ER-Model (Cont)

Chapter 2

Database Queries

- Data Definition Language
  1. CREATE DATABASE bankdb;
  2. DROP DATABASE bankdb;
  3. BACKUP DATABASE bankdb;
  4. BACKUP DATABASE bankdb
     TO DISK = ‘D:\xyz\bankdb.bak’;
  5. BACKUP DATABASE bankdb
     TO DISK = ‘D:\xyz\bankdb.bak’
     WITH DIFFERENTIAL;

Note:- do not run these queries on your machine.
It is only for specialized persons in industry

Database Queries

- DDL (Design structure or schema)
  1. CREATE TABLE cust(cust_id number(5),
     cust_nm varchar2(10)
     cust_city varchar2(10));

Column Name	Data Type	Size
Cust_id	number	5
Cust_nm	varchar2	10
cust_city	varchar2	10

Database Queries

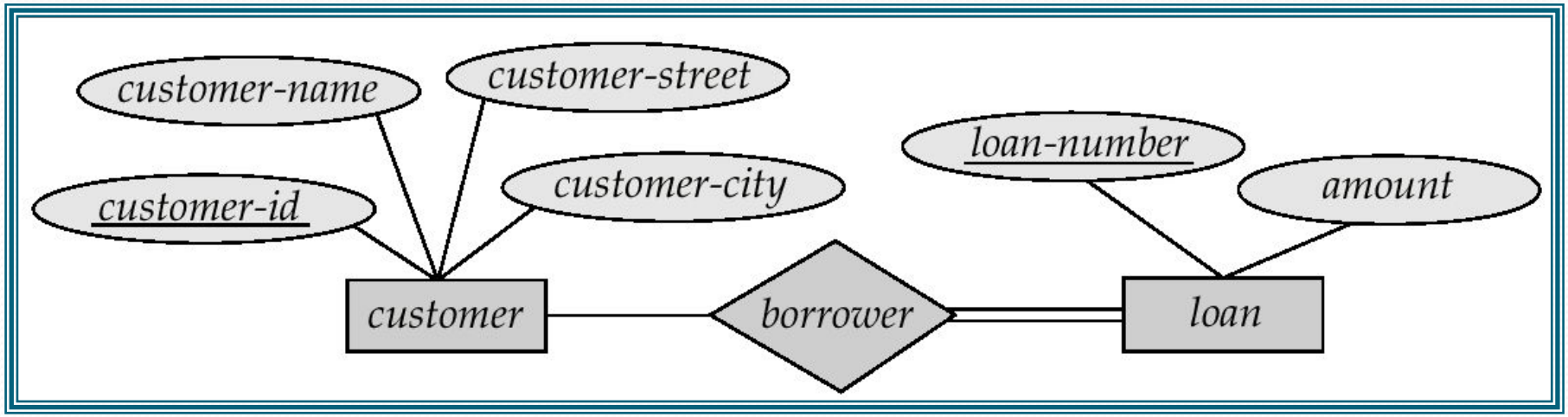
1. INSERT INTO cust(cust_id, cust_nm, cust_city)
   VALUES(101, 'abc', 'abc street');
2. SELECT * FROM cust;

Cust_id	Cust_nm	Cust_city
101	abc	abc street

3. INSERT INTO cust(cust_id, cust_nm, cust_city)
   VALUES(102, 'def', 'def street');
4. SELECT * FROM cust;

Cust_id	Cust_nm	Cust_city
101	abc	abc street
102	def	def street

Reduction of an E-R Schema to Tables



Cust_id	Cust_nm	Cust_st	Cust_ct
101	abc	Abc street	amd
102	def	Def street	srt
103	pqr	Pqr street	brd

customer

Loan_no	amt
L1	10000
L2	20000
L3	30000

loan

Cust_id	Loan_no
101	L1
102	L2
103	L3

Borrower

Reduction of an E-R Schema to Tables

- *Primary keys* allow *entity sets* and *relationship sets* to be expressed uniformly as *tables* which represent the contents of the database.
- A database which conforms to an E-R diagram can be represented by a collection of tables.
- For each entity set and relationship set there is a unique table which is assigned the name of the corresponding entity set or relationship set.
- Each table has a number of columns (generally corresponding to attributes), which have unique names.
- Converting an E-R diagram to a table format is the basis for deriving a relational database design from an E-R diagram.

Representing Entity Sets as Tables

- A strong entity set reduces to a table with the same attributes.

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

Composite and Multivalued Attributes

- Composite attributes are flattened out by creating a separate attribute for each component attribute
  - E.g. given entity set *customer* with composite attribute *name* with component attributes *first-name* and *last-name* the table corresponding to the entity set has two attributes *name.first-name* and *name.last-name*

Cust_id	Cust_nm.fnm	Cust_nm.lnm	Cust_st	Cust_ct
101	abc	Lmn	Abc street	amd
102	def	Xyz	Def street	srt
103	pqr	xyz	Pqr street	brd

Composite and Multivalued Attributes

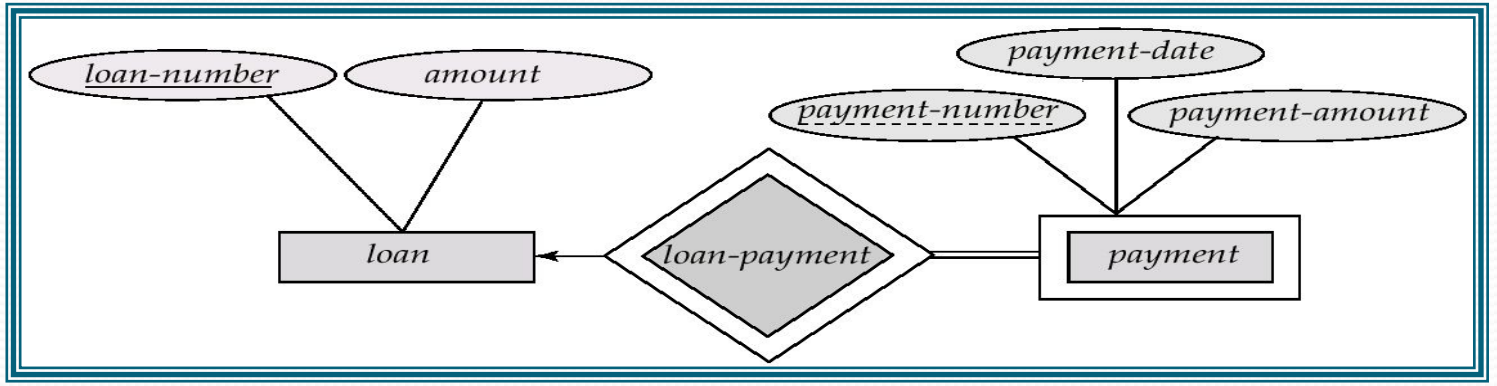
- A multivalued attribute M of an entity E is represented by a separate table EM
  - Table EM has attributes corresponding to the primary key of E and an attribute corresponding to multivalued attribute M
  - E.g. Multivalued attribute *dependent-names* of *employee* is represented by a table
    employee-dependent-names(*employee-id, dname*)
  - Each value of the multivalued attribute maps to a separate row of the table EM

emp_id	dname
101	abc
101	def
102	pqr

Employee dependent table

Representing Weak Entity Sets

- A weak entity set becomes a table that includes a column for the primary key of the identifying strong entity set



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

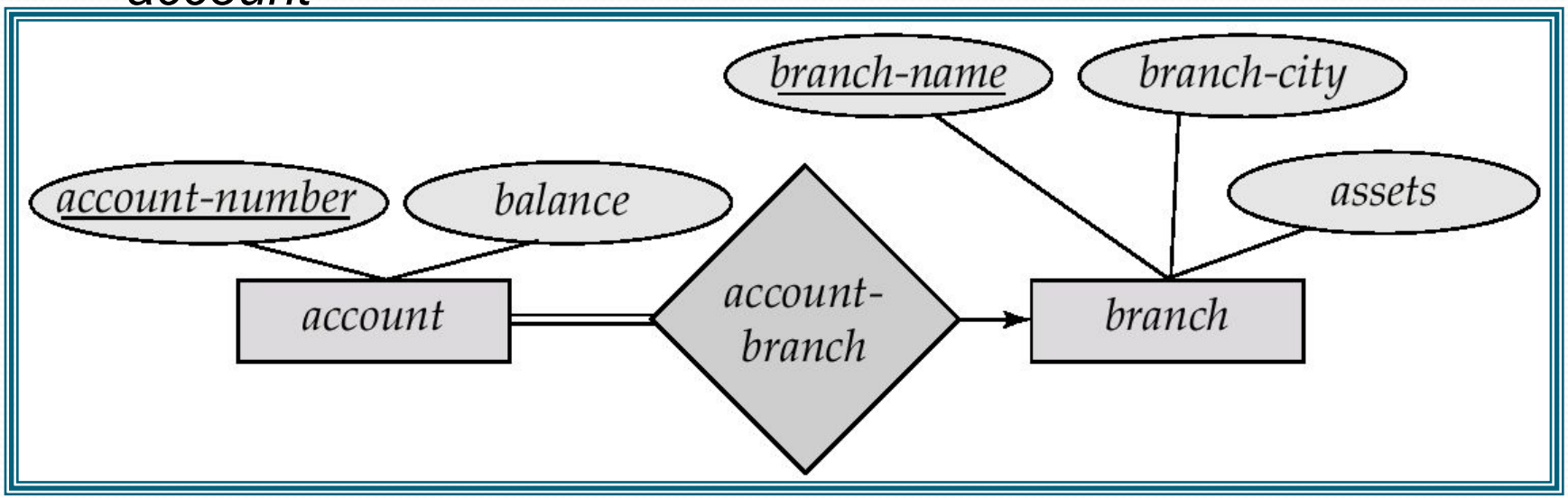
Relationship Sets as Tables

- A many-to-many relationship set is represented as a table with columns for the primary keys of the two participating entity sets, and any descriptive attributes of the relationship set.
- E.g.: table for relationship set *borrower*

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

Redundancy of Tables

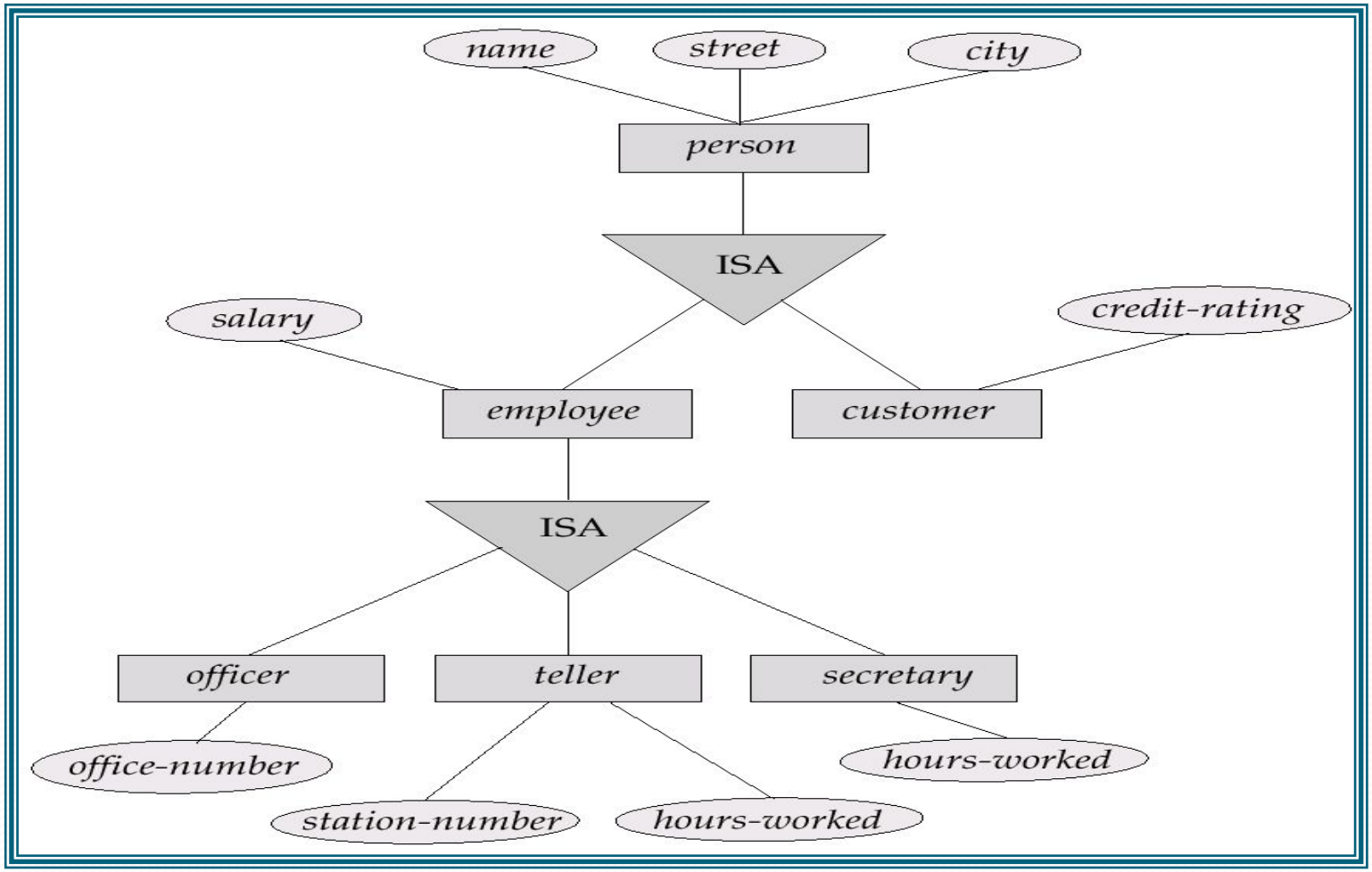
- Many-to-one and one-to-many relationship sets that are total on the many-side can be represented by adding an extra attribute to the many side, containing the primary key of the one side
- E.g.: Instead of creating a table for relationship *account-branch*, add an attribute *branch* to the entity set *account*



Redundancy of Tables

- For one-to-one relationship sets, either side can be chosen to act as the “many” side
  - That is, extra attribute can be added to either of the tables corresponding to the two entity sets
- If participation is *partial* on the many side, replacing a table by an extra attribute in the relation corresponding to the “many” side could result in null values
- The table corresponding to a relationship set linking a weak entity set to its identifying strong entity set is redundant.
  - E.g. The *payment* table already contains the information that would appear in the *loan-payment* table (i.e., the columns loan-number and *payment-number*).

Representing Specialization as Tables



Representing Specialization as Tables

- Method 1:
  - Form a table for the higher level entity
  - Form a table for each lower level entity set, include primary key of higher level entity set and local attributes

table	table attributes
person	*name, street, city*
customer	*name, credit-rating*
employee	*name, salary*

  - Drawback: getting information about, e.g., *employee* requires accessing two tables

Representing Specialization as Tables

- Method 2:
  - Form a table for each entity set with all local and inherited attributes

table	table attributes
person	*name, street, city*
customer	*name, street, city, credit-rating*
employee	*name, street, city, salary*

  - If specialization is total, table for generalized entity (*person*) not required to store information
    - Can be defined as a "view" relation containing union of specialization tables
    - But explicit table may still be needed for foreign key constraints
  - Drawback: street and city may be stored redundantly for persons who are both customers and employees

Relations Corresponding to Aggregation

- To represent aggregation, create a table containing
  - primary key of the aggregated relationship,
  - the primary key of the associated entity set
  - Any descriptive attributes

Relations Corresponding to Aggregation

- E.g. to represent aggregation *manages* between relationship *works-on* and entity set *manager*, create a table
  manages(*employee-id, branch-name, title, manager-name*)
- Table *works-on* is redundant **provided** we are willing to store null values for attribute *manager-name* in table *manages*