

# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 7

<https://powcoder.com>: Data Definition

Add WeChat powcoder

# Learning Objectives (1 of 2)

- 7.1 Data types supported by SQL standard.
- 7.2 Purpose of integrity enhancement feature of SQL.  
**Assignment Project Exam Help**
- 7.3 How to define integrity constraints using SQL.  
**<https://powcoder.com>**
- 7.4 How to use the integrity enhancement feature in the CREATE and ALTER TABLE statements  
**Add WeChat powcoder**

# Learning Objectives (2 of 2)

7.5 Purpose of views.

7.6 How to create and delete views using SQL.

Assignment Project Exam Help

7.7 How the DBMS performs operations on views.

<https://powcoder.com>

7.8 Under what conditions views are updatable.

Add WeChat powcoder

7.9 Advantages and disadvantages of views.

7.10 How the ISO transaction model works.

7.11 How to use the GRANT and REVOKE statements as a level of security.

# Table 7.1 ISO SQL Data Types

Data Type	Declarations					
boolean	BOOLEAN					
character	CHAR VARYING					
bit <sup>†</sup>	BIT	BIT VARYING				
exact numeric	NUMERIC	DECIMAL	INTEGER	SMALLINT	BIGINT	
approximate numeric	FLOAT	REAL	DOUBLE PRECISION			
datetime	DATE	TIME	TIMESTAMP			
interval	INTERVAL					
large objects	CHARACTER LARGE OBJECT		BINARY LARGE OBJECT			

<sup>†</sup>BIT and BIT VARYING have been removed from the SQL:2003 standard.

# Integrity Enhancement Feature (1 of 4)

- Consider five types of integrity constraints:
  - required data
  - domain constraints
  - entity integrity
  - referential integrity
  - general constraints.

# Integrity Enhancement Feature (2 of 4)

## Required Data

position VARCHAR(10) NOT NULL  
Assignment Project Exam Help

## Domain Constraints

<https://powcoder.com>

(a) **CHECK** Add WeChat powcoder

sex CHAR NOT NULL

CHECK (sex IN ('M', 'F'))

# Integrity Enhancement Feature (3 of 4)

## (b) CREATE DOMAIN

CREATE DOMAIN DomainName [AS] dataType  
[DEFAULT defaultOption]  
[CHECK (searchCondition)]

Add WeChat powcoder

For example:

```
CREATE DOMAIN SexType AS CHAR
    CHECK (VALUE IN ('M', 'F'));
sex SexType NOT NULL
```

# Integrity Enhancement Feature (4 of 4)

- **searchCondition** can involve a table lookup:

```
CREATE DOMAIN BranchNo AS CHAR(4)
Assignment Project Exam Help
CHECK (VALUE IN (SELECT branchNo
                  https://powcoder.com
                  FROM Branch));
```

- Domains can be removed using **DROP DOMAIN**:

```
DROP DOMAIN DomainName
[RESTRICT | CASCADE]
```

# IEF - Entity Integrity

- Primary key of a table must contain a unique, non-null value for each row.
- ISO standard supports FOREIGN KEY clause in CREATE and ALTER TABLE statements:  
<https://powcoder.com>

PRIMARY KEY(staffNo)

Add WeChat powcoder

PRIMARY KEY(clientNo, propertyNo)

- Can only have one PRIMARY KEY clause per table. Can still ensure uniqueness for alternate keys using UNIQUE:

UNIQUE(telNo)

# IEF - Referential Integrity (1 of 4)

- FK is column or set of columns that links each row in child table containing foreign FK to row of parent table containing matching PK.
- Referential integrity means that, if FK contains a value, that value must refer to existing row in parent table.
- ISO standard supports definition of FKs with FOREIGN KEY clause in CREATE and ALTER TABLE:  
  
FOREIGN KEY(branchNo) REFERENCES Branch

## IEF - Referential Integrity (2 of 4)

- Any INSERT/UPDATE attempting to create FK value in child table without matching CK value in parent is rejected. [Assignment Project Exam Help](https://powcoder.com)
- Action taken attempting to update/delete a CK value in parent table with matching rows in child is dependent on **referential action** specified using ON UPDATE and ON DELETE subclauses:
  - CASCADE - SET NULL
  - SET DEFAULT - NO ACTION

## IEF - Referential Integrity (3 of 4)

**CASCADE:** Delete row from parent and delete matching rows in child, and so on in cascading manner.

**SET NULL:** Delete row from parent and set FK column(s) in child to NULL. Only valid if FK columns are NOT NULL.  
<https://powcoder.com>

**SET DEFAULT:** Delete row from parent and set each component of FK in child to specified default. Only valid if DEFAULT specified for FK columns.

**NO ACTION:** Reject delete from parent. Default.

# IEF - Referential Integrity (4 of 4)

FOREIGN KEY (staffNo) REFERENCES Staff

ON DELETE SET NULL

Assignment Project Exam Help

FOREIGN KEY (ownerNo) REFERENCES Owner

<https://powcoder.com>

ON UPDATE CASCADE

Add WeChat powcoder

## IEF - General Constraints (1 of 2)

- Could use CHECK/UNIQUE in CREATE and ALTER TABLE.
- Similar to the ~~Assignment Project Exam Help~~ CHECK clause, also have:

<https://powcoder.com>  
CREATE ASSERTION AssertionName

CHECK (search ~~Add WeChat~~ Condition) powcoder

## IEF - General Constraints (2 of 2)

CREATE ASSERTION StaffNotHandlingTooMuch  
CHECK (NOT EXISTS (SELECT staffNo

Assignment Project Exam Help  
FROM PropertyForRent

GROUP BY staffNo  
<https://powcoder.com>

HAVING COUNT(\*) > 100))

Add WeChat powcoder

# Data Definition (1 of 2)

- SQL DDL allows database objects such as schemas, domains, tables, views, and indexes to be created and destroyed. **Assignment Project Exam Help**
- Main SQL DDL statements are:  
<https://powcoder.com>

CREATE SCHEMA

DROP SCHEMA

CREATE/ALTER DOMAIN

DROP DOMAIN

CREATE/ALTER TABLE

DROP TABLE

CREATE VIEW

DROP VIEW

- Many DBMSs also provide:

CREATE INDEX

DROP INDEX

## Data Definition (2 of 2)

- Relations and other database objects exist in an **environment**.
- Each environment contains one or more **catalogs**, and each catalog consists of set of schemas.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- Schema is named collection of related database objects.  
[Add WeChat powcoder](#)
- Objects in a schema can be tables, views, domains, assertions, collations, translations, and character sets. All have same owner.

# CREATE SCHEMA

CREATE SCHEMA [Name |

AUTHORIZATION CreatorId ]

DROP SCHEMA Name [RESTRICT | CASCADE]

- With RESTRICT (default), schema must be empty or operation fails.
- With CASCADE, operation cascades to drop all objects associated with schema in order defined above. If any of these operations fail, DROP SCHEMA fails.

# CREATE TABLE (1 of 2)

```
CREATE TABLE TableName  
{  
    (colName dataType [NOT NULL] [UNIQUE]  
     [DEFAULT defaultOption]  
     [CHECK searchCondition] [, ...])  
    [PRIMARY KEY (listOfColumns),]  
    {[UNIQUE (listOfColumns), [, ...],]}  
    {[FOREIGN KEY (listOfFKColumns)  
      REFERENCES ParentTableName [(listOfCKColumns)],  
      [ON UPDATE referentialAction]  
      [ON DELETE referentialAction]] [, ...]}  
    {[CHECK (searchCondition)] [, ...]})
```

## CREATE TABLE (2 of 2)

- Creates a table with one or more columns of the specified **dataType**.
- With NOT NULL, system rejects any attempt to insert a null in the column.  
<https://powcoder.com>
- Can specify a DEFAULT value for the column.  
[Add WeChat powcoder](#)
- Primary keys should always be specified as NOT NULL.
- FOREIGN KEY clause specifies FK along with the referential action.

## Example 7.1 - CREATE TABLE (1 of 2)

```
CREATE DOMAIN OwnerNumber AS VARCHAR(5)
    CHECK (VALUE IN (SELECT ownerNo FROM PrivateOwner));
CREATE DOMAIN StaffNumber AS VARCHAR(5)
    CHECK (VALUE IN (SELECT staffNo FROM Staff));
CREATE DOMAIN PNumber AS VARCHAR(5);
CREATE DOMAIN PRooms AS SMALLINT
    CHECK(VALUE BETWEEN 1 AND 15);
CREATE DOMAIN PRent AS DECIMAL(6,2)
    CHECK(VALUE BETWEEN 0 AND 9999.99);
```

## Example 7.1 - CREATE TABLE (2 of 2)

```
CREATE TABLE PropertyForRent (
    propertyNo Pnumber NOT NULL, ....
    rooms PRooms NOT NULL DEFAULT 1,
    rent PRent NOT NULL, DEFAULT 600,
    ownerNo OwnerNumber NOT NULL,
    staffNo StaffNumber
        Constraint StaffNotHandlingTooMuch ....
    branchNo BranchNumber NOT NULL,
    PRIMARY KEY (propertyNo),
    FOREIGN KEY (staffNo) REFERENCES Staff
        ON DELETE SET NULL ON UPDATE CASCADE ....);
```

# ALTER TABLE

- Add a new column to a table.
- Drop a column from a table.

Assignment Project Exam Help

- Add a new table constraint.

<https://powcoder.com>

- Drop a table constraint.

Add WeChat powcoder

- Set a default for a column.

- Drop a default for a column.

## Example 7.2(a) – ALTER TABLE

Change Staff table by removing default of ‘Assistant’ for position column and setting default for sex column to female ('F'). **Assignment Project Exam Help**

ALTER TABLE Staff

ALTER position DROP DEFAULT;  
ALTER TABLE Staff

ALTER sex SET DEFAULT 'F';

## Example 7.2(b) – ALTER TABLE

Remove constraint from PropertyForRent that staff are not allowed to handle more than 100 properties at a time. Add new column to Client table

**Assignment Project Exam Help**

ALTER TABLE PropertyForRent  
<https://powcoder.com>

DROP CONSTRAINT StaffNotHandlingTooMuch;  
**Add WeChat powcoder**

ALTER TABLE Client

ADD prefNoRooms PRooms;

# DROP TABLE

DROP TABLE TableName [RESTRICT | CASCADE]

e.g. DROP TABLE PropertyForRent;  
**Assignment Project Exam Help**

- Removes named [table](https://powcoder.com) and all rows within it.
- With RESTRICT, if any other objects depend for their existence on continued existence of this table, SQL does not allow request.
- With CASCADE, SQL drops all dependent objects (and objects dependent on these objects).

# Views (1 of 2)

## View

Dynamic result of one or more relational operations  
operating on base relations to produce another relation.

- Virtual relation that does not necessarily actually exist in the database but is produced upon request, at time of request.

## Views (2 of 2)

- Contents of a view are defined as a query on one or more base relations.
- With **view resolution**, any operations on view are automatically translated into operations on relations from which it is derived.
- With **view materialization**, the view is stored as a temporary table, which is maintained as the underlying base tables are updated.

# SQL - CREATE VIEW (1 of 2)

CREATE VIEW ViewName [ (newColumnName [...]) ]  
AS subselect

[WITH [CASCADED | LOCAL] CHECK OPTION]

- Can assign a name to each column in view.
- If list of column names is specified, it must have same number of items as number of columns produced by **subselect**.
- If omitted, each column takes name of corresponding column in **subselect**.

## SQL - CREATE VIEW (2 of 2)

- List must be specified if there is any ambiguity in a column name.
- The **subselect** is known as the **defining query**.
- WITH CHECK OPTION ensures that if a row fails to satisfy WHERE clause of defining query, it is not added to underlying base table.
- Need SELECT privilege on all tables referenced in subselect and USAGE privilege on any domains used in referenced columns.

## Example 7.3 - Create Horizontal View

Create view so that manager at branch B003 can only see details for staff who work in his or her office.

Assignment Project Exam Help

```
CREATE VIEW Manager3Staff
AS SELECT *
FROM Staff
WHERE branchNo = 'B003';
```

Add WeChat powcoder

staffNo	fName	lName	position	sex	DOB	salary	branchNo
SG37	Ann	Beech	Assistant	F	10-Nov-60	12000.00	B003
SG14	David	Ford	Supervisor	M	24-Mar-58	18000.00	B003
SG5	Susan	Brand	Manager	F	3-Jun-40	24000.00	B003

## Example 7.4 - Create Vertical View

Create view of staff details at branch B003 excluding salaries.

```
CREATE VIEW Staff3  
AS SELECT staffNo, fName, lName, position, sex  
FROM Staff  
WHERE branchNo = 'B003';
```

staffNo	fName	lName	position	sex
SG37	Ann	Beech	Assistant	F
SG14	David	Ford	Supervisor	M
SG5	Susan	Brand	Manager	F

## Example 7.5 - Grouped and Joined Views

Create view of staff who manage properties for rent,  
including branch number they work at, staff number, and  
number of properties they manage

**Assignment Project Exam Help**

```
CREATE VIEW StaffPropCnt (branchNo, staffNo, cnt)
AS SELECT s.branchNo, s.staffNo, COUNT(*)
        FROM Staff s, PropertyForRent p
       WHERE s.staffNo = p.staffNo
      GROUP BY s.branchNo, s.staffNo;
```

# Example 7.3 - Grouped and Joined Views

branchNo	staffNo	cnt
B002	SG14	1
B003	SG37	2
B005	SL41	1
B007	SA9	1

# SQL - DROP VIEW (1 of 2)

DROP VIEW ViewName [RESTRICT | CASCADE]

- Causes definition of view to be deleted from database.
- For example:

<https://powcoder.com>

DROP VIEW Manager3Staff;

Add WeChat powcoder

## SQL - DROP VIEW (2 of 2)

- With CASCADE, all related dependent objects are deleted; i.e. any views defined on view being dropped.
- With RESTRICT (default), if any other objects depend for their existence on continued existence of view being dropped, command is rejected.

Add WeChat powcoder

# View Resolution (1 of 4)

Count number of properties managed by each member at branch B003.

Assignment Project Exam Help  
SELECT staffNo, cnt  
FROM StaffPropCnt  
WHERE branchNo = 'B003'  
ORDER BY staffNo;

## View Resolution (2 of 4)

- (a) View column names in SELECT list are translated into their corresponding column names in the defining query:

Assignment Project Exam Help

SELECT s.staffNo As staffNo, COUNT(\*) As cnt  
https://powcoder.com

- (b) View names in FROM are replaced with corresponding FROM lists of defining query:  
*Add WeChat powcoder*

FROM Staff s, PropertyForRent p

## View Resolution (3 of 4)

- (c) WHERE from user query is combined with WHERE of defining query using AND:

WHERE s.staffNo = p.staffNo AND branchNo = 'B003'

- (d) GROUP BY and HAVING clauses copied from defining query:

Add WeChat powcoder

GROUP BY s.branchNo, s.staffNo

- (e) ORDER BY copied from query with view column name translated into defining query column name

ORDER BY s.staffNo

## View Resolution (4 of 4)

(f) Final merged query is now executed to produce the result:

```
Assignment Project Exam Help  
SELECT s.staffNo AS staffNo, COUNT(*) AS cnt  
FROM Staff s,PropertyForRent p  
WHERE s.staffNo = p.staffNo AND  
      branchNo = 'B003'  
      Add WeChat powcoder  
GROUP BY s.branchNo, s.staffNo  
ORDER BY s.staffNo;
```

# Restrictions on Views (1 of 3)

SQL imposes several restrictions on creation and use of views.

- (a) If column in view is based on an aggregate function:
- Column may appear only in SELECT and ORDER BY clauses of queries that access view.
  - Column may not be used in WHERE nor be an argument to an aggregate function in any query based on view.

## Restrictions on Views (2 of 3)

- For example, following query would fail:

```
SELECT COUNT(cnt)
FROM StaffPropCnt;
```

<https://powcoder.com>

- Similarly, following query would also fail:

[Add WeChat](#) [powcoder](#)

```
SELECT *
FROM StaffPropCnt
WHERE cnt > 2;
```

## Restrictions on Views (3 of 3)

- (b) Grouped view may never be joined with a base table or a view.

Assignment Project Exam Help

- For example, StaffPropCnt view is a grouped view, so any attempt to join this view with another table or view fails.

Add WeChat powcoder

# View Updatability (1 of 5)

- All updates to base table reflected in all views that encompass base table.
- Similarly, ~~Assignment Project Exam Help~~ may expect that if view is updated then base table(s) will reflect change.  
<https://powcoder.com>

Add WeChat powcoder

## View Updatability (2 of 5)

- However, consider again view StaffPropCnt.
- If we tried to insert record showing that at branch B003, S G5 manages 2 properties:  
**Assignment Project Exam Help**

**https://powcoder.com**  
INSERT INTO StaffPropCnt  
VALUES ('B003', 'SG5', 2);  
**Add WeChat powcoder**

- Have to insert 2 records into PropertyForRent showing which properties SG5 manages. However, do not know which properties they are; i.e. do not know primary keys!

## View Updatability (3 of 5)

- If change definition of view and replace count with actual property numbers:

Assignment Project Exam Help  
<https://powcoder.com>

```
CREATE VIEW StaffPropList (branchNo,
                            staffNo, propertyNo)
AS SELECT s.branchNo, s.staffNo, p.propertyNo
FROM Staff s, PropertyForRent p
WHERE s.staffNo = p.staffNo;
```

## View Updatability (4 of 5)

- Now try to insert the record:

```
INSERT INTO StaffPropList  
VALUES ('B003', 'SG5', 'PG19');
```

<https://powcoder.com>

- Still problem, because in PropertyForRent all columns except postcode/staffNo are not allowed nulls.
- However, have no way of giving remaining non-null columns values.

# View Updatability (5 of 5)

- ISO specifies that a view is updatable if and only if:
  - DISTINCT is not specified.
  - Every element in SELECT list of defining query is a column name and no column appears more than once.
  - FROM clause specifies only one table, excluding any views based on a join, union, intersection or difference.
  - No nested SELECT referencing outer table.
  - No GROUP BY or HAVING clause.
  - Also, every row added through view must not violate integrity constraints of base table.

# Updatable View

For view to be updatable, DBMS must be able to trace any row or column back to its row or column in the source table.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# WITH CHECK OPTION (1 of 2)

- Rows exist in a view because they satisfy WHERE condition of defining query.
- If a row changes and no longer satisfies condition, it disappears from the view.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- New rows appear within view when insert/update on view cause them to satisfy WHERE condition.  
[Add WeChat powcoder](#)
- Rows that enter or leave a view are called **migrating rows**.
- WITH CHECK OPTION prohibits a row migrating out of the view.

## WITH CHECK OPTION (2 of 2)

- LOCAL/CASCADED apply to view hierarchies.
- With LOCAL, any row insert/update on view and any view directly or indirectly defined on this view must not cause row to disappear from view unless row also disappears from derived view/table.  
*Assignment Project Exam Help  
https://powcoder.com*
- With CASCDED (default), any row insert/ update on this view and on any view directly or indirectly defined on this view must not cause row to disappear from the view.  
*Add WeChat powcoder*

## Example 7.6 - WITH CHECK OPTION (1 of 4)

```
CREATE VIEW Manager3Staff  
AS SELECT *  
        FROM Staff  
       WHERE branchNo = 'B003'  
      WITH CHECK OPTION;
```

Add WeChat powcoder

- Cannot update branch number of row B003 to B002 as this would cause row to migrate from view.
- Also cannot insert a row into view with a branch number that does not equal B003.

## Example 7.6 - WITH CHECK OPTION (2 of 4)

- Now consider the following:

```
CREATE VIEW LowSalary  
AS SELECT * FROM Staff WHERE salary > 9000;  
CREATE VIEW HighSalary  
AS SELECT * FROM LowSalary  
      WHERE salary > 10000  
WITH LOCAL CHECK OPTION;  
CREATE VIEW Manager3Staff  
AS SELECT * FROM HighSalary  
      WHERE branchNo = 'B003';
```

## Example 7.6 - WITH CHECK OPTION (3 of 4)

```
UPDATE Manager3Staff
```

```
SET salary = 9500
```

```
WHERE status = SG37;
```

- This update would fail: although update would cause row to disappear from HighSalary, row would not disappear from LowSalary.
- However, if update tried to set salary to 8000, update would succeed as row would no longer be part of LowSalary.

## Example 7.6 - WITH CHECK OPTION (4 of 4)

- If HighSalary had specified WITH CASCADED CHECK OPTION, setting salary to 9500 or 8000 would be rejected because row would disappear from HighSalary.
- To prevent anomalies like this, each view should be created using WITH CASCADED CHECK OPTION.

Add WeChat powcoder

# Advantages of Views

- Data independence
- Currency  
Assignment Project Exam Help  
<https://powcoder.com>
- Improved security
- Reduced complexity
- Convenience Add WeChat powcoder
- Customization
- Data integrity

# Disadvantages of Views

- Update restriction
- Structure restriction
- Performance

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# View Materialization (1 of 2)

- View resolution mechanism may be slow, particularly if view is accessed frequently.
- View materialization stores view as temporary table when view is first queried.  
<https://powcoder.com>
- Thereafter, queries based on materialized view can be faster than recomputing view each time.
- Difficulty is maintaining the currency of view while base tables(s) are being updated.

# View Maintenance

- **View maintenance** aims to apply only those changes necessary to keep view current.
- Consider following view:

```
CREATE VIEW StaffPropRent(staffNo)
AS SELECT DISTINCT staffNo
          Add WeChat powcoder
          FROM PropertyForRent
          WHERE branchNo = 'B003' AND
              rent > 400;
```

staffNo
SG37
SG14

## View Materialization (2 of 2)

- If insert row into PropertyForRent with rent  $\leq$  400 then view would be unchanged.
- If insert row for property PG24 at branch B003 with staffNo = SG19 and rent = 550, then row would appear in materialized view.
- If insert row for property PG54 at branch B003 with staffNo = SG37 and rent = 450, then no new row would need to be added to materialized view.
- If delete property PG24, row should be deleted from materialized view.
- If delete property PG54, then row for PG37 should not be deleted (because of existing property PG21).

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Transactions (1 of 3)

- SQL defines transaction model based on COMMIT and ROLL BACK.
- Transaction is a logical unit of work with one or more SQL statements guaranteed to be atomic with respect to recovery.  
<https://powcoder.com>
- An SQL transaction automatically begins with a **transaction-initiating** SQL statement (e.g., SELECT, INSERT).
- Changes made by transaction are not visible to other concurrently executing transactions until transaction completes.

## Transactions (2 of 3)

- Transaction can complete in one of four ways:
  - COMMIT ends transaction successfully, making changes
  - ROLLBACK aborts transaction, backing out any changes made by transaction.
  - For programmatic SQL, successful program termination ends final transaction successfully, even if COMMIT has not been executed.
  - For programmatic SQL, abnormal program end aborts transaction.

# Transactions (3 of 3)

- New transaction starts with next transaction-initiating statement.
- SQL transactions cannot be nested.
- SET TRANSACTION configures transaction:

SET TRANSACTION [READ ONLY | READ WRITE] |  
[ISOLATION LEVEL READ UNCOMMITTED |  
READ COMMITTED | REPEATABLE READ |  
SERIALIZABLE]

# Immediate and Deferred Integrity Constraints (1 of 2)

- Do not always want constraints to be checked immediately, but instead at transaction commit.
- Constraint may be defined as INITIALLY IMMEDIATE or INITIALLY DEFERRED, indicating mode the constraint assumes at start of each transaction.  
<https://powcoder.com>
- In former case, also possible to specify whether mode can be changed subsequently using qualifier [NOT] DEFERRABLE.  
[Add WeChat powcoder](#)
- Default mode is INITIALLY IMMEDIATE.

# Immediate and Deferred Integrity Constraints (2 of 2)

- SET CONSTRAINTS statement used to set mode for specified constraints for current transaction:

Assignment Project Exam Help

SET CONSTRAINTS

<https://powcoder.com>  
{ALL | constraintName [, . . . ]}

{DEFERRED | IMMEDIATE}

# Access Control - Authorization Identifiers and Ownership

- Authorization identifier is normal SQL identifier used to establish identity of a user. Usually has an associated password. [Assignment Project Exam Help](https://powcoder.com)
- Used to determine which objects user may reference and what operations may be performed on those objects.
- Each object created in SQL has an owner, as defined in [Authorization](https://powcoder.com) clause of schema to which object belongs.
- Owner is only person who may know about it.

# Privileges (1 of 2)

- Actions user permitted to carry out on given base table or view:

SELECT Retrieve data from a table.

INSERT Insert new rows into a table.

UPDATE Modify rows of data in a table.

DELETE Delete rows of data from a table.

REFERENCES Reference columns of named table in integrity constraints.

USAGE Use domains, collations, character sets, and translations.

## Privileges (2 of 2)

- Can restrict INSERT/UPDATE/REFERENCES to named columns.
- Owner of table must grant other users the necessary privileges using GRANT statement.  
<https://powcoder.com>
- To create view, user must have SELECT privilege on all tables that make up view and REFERENCES privilege on the named columns.

# GRANT (1 of 2)

```
GRANT {PrivilegeList | ALL PRIVILEGES}
ON ObjectName
TO{AuthorizationIDList | PUBLIC}
[WITH GRANT OPTION]
```

<https://powcoder.com>

- **PrivilegeList** consists of one or more of above privileges separated by commas.
- **ALL PRIVILEGES** grants all privileges to a user.

## GRANT (2 of 2)

- PUBLIC allows access to be granted to all present and future authorized users.
- **ObjectName** can be a base table, view, domain, character set, collation or translation.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- WITH GRANT OPTION allows privileges to be passed on.  
[Add WeChat powcoder](#)

## Example 7.7-7.8 - GRANT

Give Manager full privileges to Staff table.

```
GRANT ALL PRIVILEGES  
ON Staff  
TO Manager WITH GRANT OPTION;
```

Give users Personnel and Director SELECT and UPDATE  
on column salary of Staff.

```
GRANT SELECT, UPDATE (salary)  
ON Staff  
TO Personnel, Director;
```

# Example 7.9 - GRANT Specific Privileges to PUBLIC

Give all users SELECT on Branch table.

```
GRANT SELECT  
ON Branch  
TO PUBLIC;
```

Add WeChat powcoder

# REVOKE (1 of 3)

- REVOKE takes away privileges granted with GRANT.

```
REVOKE [GRANT OPTION FOR]
{PrivilegeList | ALL PRIVILEGES}
ON ObjectName
FROM {AuthorizationIdList | PUBLIC}
[RESTRICT | CASCADE]
```

- ALL PRIVILEGES refers to all privileges granted to a user by user revoking privileges.

## REVOKE (2 of 3)

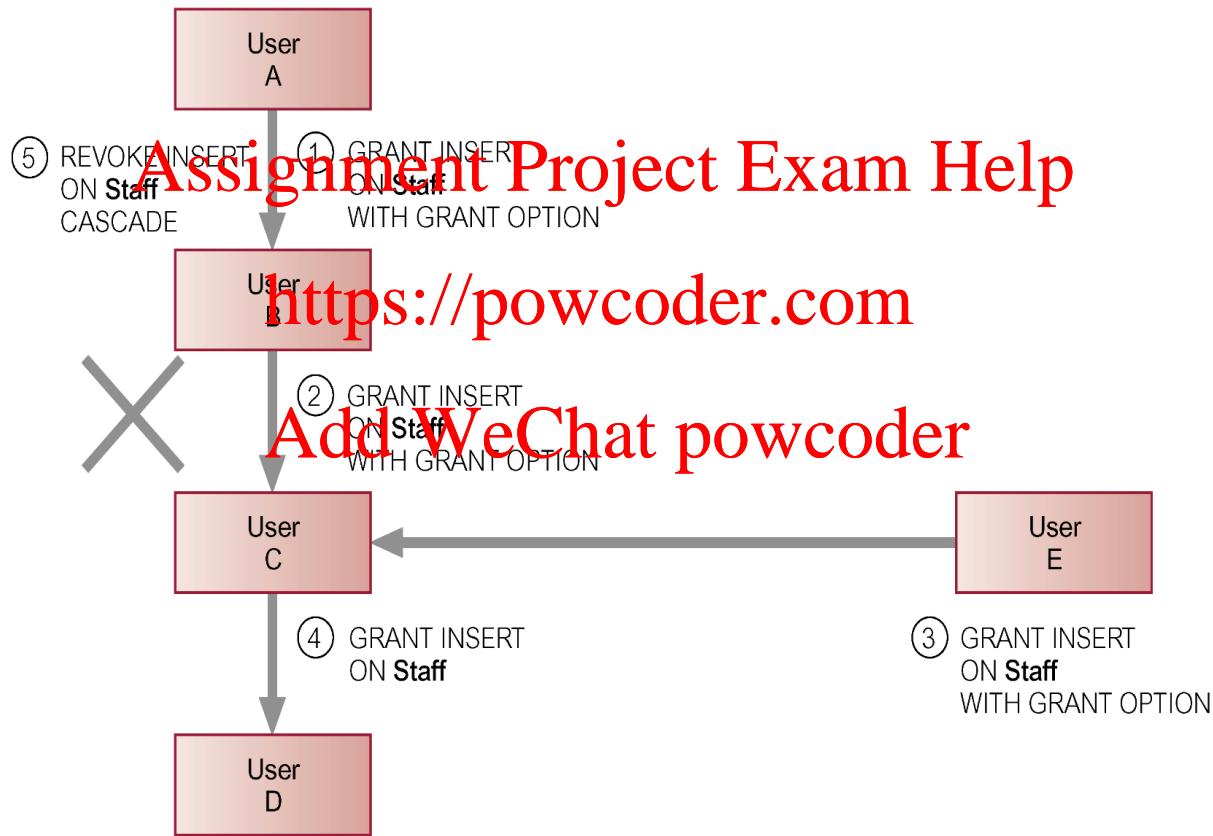
- GRANT OPTION FOR allows privileges passed on via WITH GRANT OPTION of GRANT to be revoked separately from the privileges themselves.
- REVOKE fails if it results in an abandoned object, such as a view, unless the CASCADE keyword has been specified.
- Privileges granted to this user by other users are not affected.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# REVOKE (3 of 3)



# Example 7.10 -7.11 - REVOKE Specific Privileges

Revoke privilege SELECT on Branch table from all users.

```
REVOKE SELECT  
Assignment Project Exam Help  
ON Branch  
FROM PUBLIC;
```

Add WeChat powcoder  
Revoke all privileges given to Director on Staff table.

```
REVOKE ALL PRIVILEGES  
ON Staff  
FROM Director;
```

# Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 8

<https://powcoder.com> Advanced SQL

Add WeChat powcoder

# Learning Objectives

- 8.1 How to use the SQL programming language
- 8.2 How to use SQL cursors  
[Assignment Project Exam Help](#)
- 8.3 How to create stored procedures  
<https://powcoder.com>
- 8.4 How to create triggers  
[Add WeChat powcoder](#)
- 8.5 How to use triggers to enforce integrity constraints
- 8.6 The advantages and disadvantages of triggers
- 8.7 How to use recursive queries

# The SQL Programming Language (1 of 2)

- Impedance mismatch
  - Mixing different programming paradigms
  - SQL is a declarative language
  - High-level language such as C is a procedural language
  - SQL and 3GLs use different models to represent data

# The SQL Programming Language (2 of 2)

- SQL/PSM (Persistent Stored Modules)
- PL/SQL (Procedural Language/SQL)
  - Oracle's procedural extension to SQL
  - Two versions <https://powcoder.com>

Add WeChat powcoder

# Declarations (1 of 2)

- Variables and constant variables must be declared before they can be referenced
- Possible to declare a variable as NOT NULL
- %TYPE – variable same type as a column
  - vStaffNo Staff.staffNo%TYPE;
- %ROWTYPE – variable same type as an entire row
  - vStaffNo1 Staff%ROWTYPE;

# Declarations (2 of 2)

[DECLARE **Assignment Project Exam Help** *Optional*  
— declarations]  
BEGIN **https://powcoder.com** *Mandatory*  
— executable statements  
**Add WeChat powcoder**  
[EXCEPTION *Optional*  
— exception handlers]  
END; *Mandatory*

# Assignments

- Variables can be assigned in two ways:
  - Using the normal assignment statement (:=):

Assignment Project Exam Help

vStaffNo := 'SG14';

<https://powcoder.com>

- Using an SQL SELECT or FETCH statement:

Add WeChat powcoder

**SELECT COUNT(\*) INTO x**

**FROM PropertyForRent**

**WHERE staffNo = vStaffNo;**

# Control Statements

- Conditional IF statement
- Conditional CASE statement
- Iteration statement (LOOP)
- Iteration statement (WHILE and REPEAT)
- Iteration statement (FOR)

# Conditional IF Statement

**IF** (position = 'Manager') **THEN**

    salary := salary\*1.05;

**ELSE**           Assignment Project Exam Help

    salary := salary\*1.05;  
<https://powcoder.com>

**END IF;**

Add WeChat powcoder

# Conditional CASE Statement

**UPDATE** Staff

**SET** salary = **CASE**

**WHEN** position = 'Manager'

**THEN** salary \* 1.05  
<https://powcoder.com>

**ELSE**

salary \* 1.02  
Add WeChat powcoder

**END;**

# Iteration Statement (LOOP)

x:=1;

myLoop:

**LOOP**

Assignment Project Exam Help

x := x+1;

**IF** (x > 3) **THEN**

<https://powcoder.com>

**EXIT** myLoop; -- exit loop now  
Add WeChat powcoder

**END LOOP** myLoop;

--- control resumes here

y := 2;

# Iteration Statement (WHILE and REPEAT)

**WHILE** (condition) **DO**

<SQL statement list>

Assignment Project Exam Help

**END WHILE** [labelName];

<https://powcoder.com>

**REPEAT**

Add WeChat powcoder

<SQL statement list>

**UNTIL** (condition)

**END REPEAT** [labelName];

# Iteration Statement (FOR)

myLoop1:

```
FOR iStaff AS SELECT COUNT(*) FROM
    Assignment Project Exam Help
PropertyForRent WHERE staffNo = 'SG14' DO
    https://powcoder.com
    ....
    Add WeChat powcoder
END FOR myLoop1;
```

# Exceptions in PL/SQL

- Exception
  - Identifier in PL/SQL
  - Raised during the execution of a block
  - Terminates block's main body of actions
- Exception handlers
  - Separate routines that handle raised exceptions
- User-defined exception
  - Defined in the declarative part of a PL/SQL block

# Example of Exception Handling in PL/SQL

```
DECLARE
    vpCount      NUMBER;
    vStaffNo PropertyForRent.staffNo%TYPE := 'SG14';
    -- define an exception for the enterprise constraint that prevents a member of staff
    -- managing more than 100 properties
    e_too_many_properties EXCEPTION;
    PRAGMA EXCEPTION_INIT(e_too_many_properties, -20000);
BEGIN
    SELECT COUNT(*) INTO vpCount
    FROM PropertyForRent
    WHERE staffNo = vStaffNo;
    IF vpCount = 100
        -- raise an exception for the general constraint
        RAISE e_too_many_properties;
    END IF;
    UPDATE PropertyForRent SET staffNo = vStaffNo WHERE propertyNo = 'PG4';
EXCEPTION
    -- handle the exception for the general constraint
    WHEN e_too_many_properties THEN
        dbms_output.put_line('Member of staff ' || staffNo || 'already managing 100 properties');
END;
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Condition Handling

- Define a handler by:
  - Specifying its type
  - Exception and completion conditions it can resolve
  - Action it takes to do so  
<https://powcoder.com>
- Handler is activated:
  - When it is the most appropriate handler for the condition that has been raised by the SQL statement

# The DECLARE . . . HANDLER Statement

```
DECLARE {CONTINUE | EXIT | UNDO} HANDLER  
FOR SQLSTATE {sqlstateValue | conditionName | SQL  
EXCEPTION|SQLWARNING|Project|Exam|Help} handler  
Action;
```

<https://powcoder.com>

Add WeChat powcoder

# Cursors in PL/SQL

- **Cursor**
  - Allows the rows of a query result to be accessed one at a time [Assignment](#) [Project](#) [Exam](#) [Help](#)
  - Must be declared and opened before use <https://powcoder.com>
  - Must be closed to deactivate it after it is no longer required [Add WeChat powcoder](#)
  - Updating rows through a cursor

# Using Cursors in PL/SQL to Process a Multirow Query

```
DECLARE
    vPropertyNo      PropertyForRent.propertyNo%TYPE;
    vStreet          PropertyForRent.street%TYPE;
    vCity            PropertyForRent.city%TYPE;
    vPostcode        PropertyForRent.postcode%TYPE;
    CURSOR propertyCursor IS
        SELECT propertyNo, street, city, postcode
        FROM PropertyForRent
        WHERE staffNo = 'SG1'
        ORDER BY propertyNo;
BEGIN
    -- Open the cursor to start of selection, then loop to fetch each row of the result table
    OPEN propertyCursor;
    LOOP
        -- Fetch next row of the result table
        FETCH propertyCursor
        INTO vPropertyNo, vStreet, vCity, vPostcode;
        EXIT WHEN propertyCursor%NOTFOUND;
        -- Display data
        dbms_output.put_line('Property number: ' || vPropertyNo);
        dbms_output.put_line('Street:           ' || vStreet);
        dbms_output.put_line('City:            ' || vCity);
        IF postcode IS NOT NULL THEN
            dbms_output.put_line('Post Code:       ' || vPostcode);
        ELSE
            dbms_output.put_line('Post Code:       NULL');
        END IF;
    END LOOP;
    IF propertyCursor%ISOPEN THEN CLOSE propertyCursor END IF;

    -- Error condition - print out error
EXCEPTION
    WHEN OTHERS THEN
        dbms_output.put_line('Error detected');
        IF propertyCursor%ISOPEN THEN CLOSE propertyCursor; END IF;
END;
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Subprograms, Stored Procedures, Functions, and Packages (1 of 2)

- **Subprograms**
  - Named PL/SQL blocks that can take parameters and be invoked
- Two types:
  - Stored procedures
  - Functions (returns a single value to caller)
- Can take a set of parameters
  - Each has name and data type
  - Can be designated as IN, OUT, IN OUT

# Subprograms, Stored Procedures, Functions, and Packages (2 of 2)

- **Package**
  - Collection of procedures, functions, variables, and SQL statements that are grouped together and stored as a single program unit  
<https://powcoder.com>
- Specification
  - Declares all public constructs of the package
- Body
  - Defines all constructs (public and private) of the package

# Triggers

- Trigger
  - Defines an action that the database should take when some event occurs in the application
  - Based on Event-Condition-Action (ECA) model  
<https://powcoder.com>
- Types
  - Row-level    [Add WeChat powcoder](#)
  - Statement-level
- Event: INSERT, UPDATE or DELETE
- Timing: BEFORE, AFTER or INSTEAD OF
- Advantages and disadvantages of triggers

# Trigger Format

**CREATE TRIGGER** TriggerName

**BEFORE | AFTER | INSTEAD OF**

**INSERT | ~~Assignment Project Exam Help~~ DELETE | UPDATE [OF TriggerColumnList]**

**ON TableName**

<https://powcoder.com>

**[REFERENCING {OLD | NEW} AS {OldName | NewName}]**

**[FOR EACH {ROW | STATEMENT}]**

**[WHEN Condition]**

**<trigger action>**

# Using a BEFORE Trigger

```
CREATE TRIGGER StaffNotHandlingTooMuch
BEFORE INSERT ON PropertyForRent
REFERENCING NEW AS newrow
FOR EACH ROW
DECLARE
    vpCount      NUMBER;
BEGIN
    SELECT COUNT(*) INTO vpCount
    FROM PropertyForRent
    WHERE staffNo = :newrow.staffNo;
    IF vpCount = 100
        raise_application_error(-20000, ('Member' || :newrow.staffNo || 'already managing 100 properties'));
    END IF;
END;
```

Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder

# Triggers – Advantages

- Elimination of redundant code
- Simplifying modifications  
[Assignment Project Exam Help](#)
- Increased security  
<https://powcoder.com>
- Improved integrity  
[Add WeChat powcoder](#)
- Improved processing power
- Good fit with client-server architecture

# Triggers – Disadvantages

- Performance overhead
- Cascading effects
- Cannot be scheduled
- Less portable

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Recursion

- Extremely difficult to handle recursive queries
  - Queries about relationships that a relation has with itself (directly or indirectly)
- WITH RECURSIVE statement handles this  
<https://powcoder.com>
- Infinite loop can occur unless the cycle can be detected
  - CYCLE clause

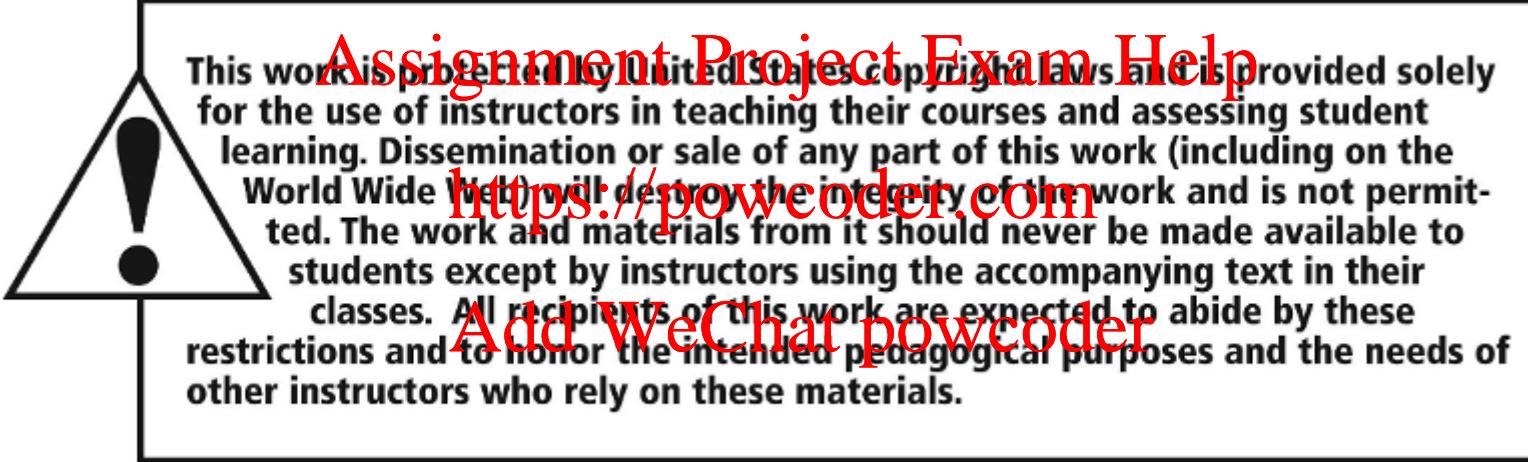
# Recursion - Example

## WITH RECURSIVE

```
AllManagers (staffNo, managerStaffNo) AS  
(SELECT staffNo, managerStaffNo  
FROM Staff  
UNION  
SELECT in.staffNo, AddWeChat powcoder  
FROM AllManagers in, Staff out  
WHERE in.managerStaffNo = out.staffNo);  
SELECT * FROM AllManagers  
ORDER BY staffNo, managerStaffNo;
```

<https://powcoder.com>

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 10

<https://powcoder.com>  
Database System  
Development Lifecycle  
Add WeChat powcoder

# Learning Objectives

- 10.1** Main components of an information system.
- 10.2** Main stages of database system development lifecycle.  
**Assignment Project Exam Help**
- 10.3** Main phases of database design: conceptual, logical, and physical design. <https://powcoder.com>
- 10.4** Benefits of CASE tools.  
**Add WeChat powcoder**
- 10.5** How to evaluate and select a DBMS.
- 10.6** Distinction between data administration and database administration.
- 10.7** Purpose and tasks associated with data administration and database administration.

# Software Depression (1 of 3)

- Last few decades have seen proliferation of software applications, many requiring constant maintenance involving: **Assignment Project Exam Help**
  - correcting faults,
  - implementing new user requirements,
  - modifying software to run on new or upgraded platforms.
- Effort spent on maintenance began to absorb resources at an alarming rate.

# Software Depression (2 of 3)

- As a result, many major software projects were
  - late,
  - over budget,
  - unreliable, <https://powcoder.com>
  - difficult to maintain,
  - performed poorly.
- In late 1960s, led to ‘software crisis’, now refer to as the ‘software depression’.

# Software Depression (3 of 3)

- Major reasons for failure of software projects includes:
  - lack of a complete requirements specification;
  - lack of appropriate development methodology;
  - poor decomposition of design into manageable components.
- Structured approach to development was proposed called Information Systems Lifecycle (ISLC).

# Information System

Resources that enable collection, management, control, and dissemination of information throughout an organization.

**Assignment Project Exam Help**

- Database is fundamental component of IS, and its development/usage should be viewed from perspective of the wider requirements of the organization.

<https://powcoder.com>

Add WeChat powcoder

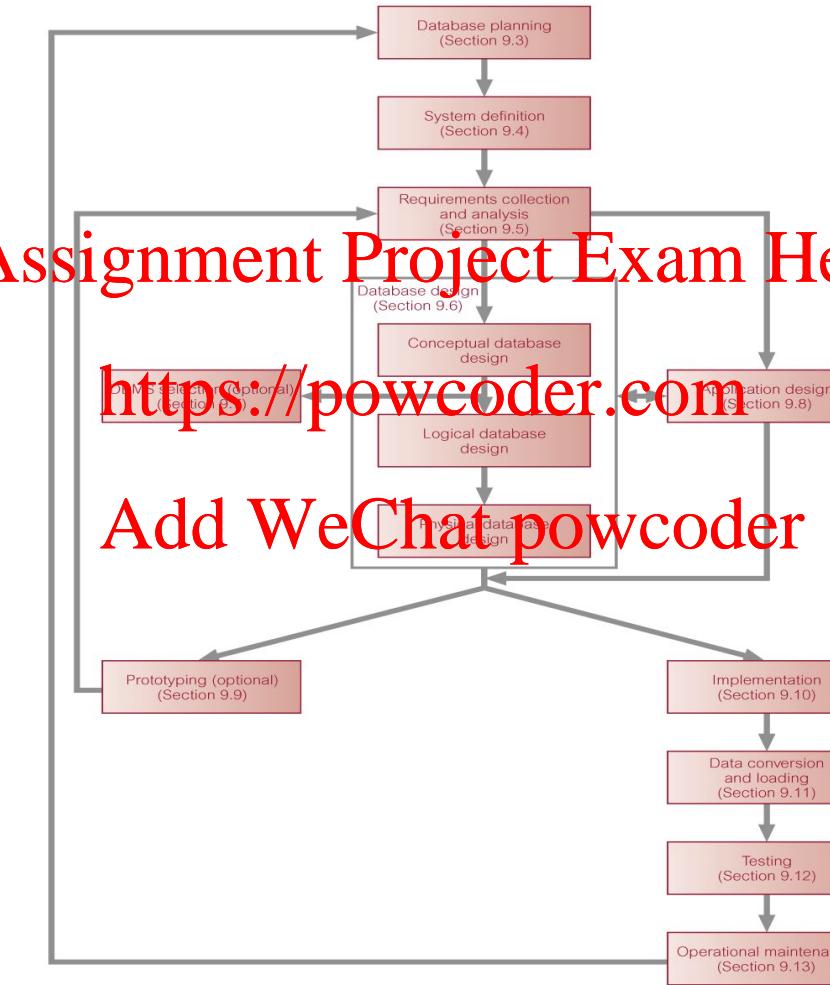
# Database System Development Lifecycle (1 of 2)

- Database planning
- System definition  
**Assignment Project Exam Help**  
**https://powcoder.com**
- Requirements collection and analysis
- Database design
- DBMS selection (optional)  
**Add WeChat powcoder**

# Database System Development Lifecycle (2 of 2)

- Application design
- Prototyping (optional)  
**Assignment Project Exam Help**  
**https://powcoder.com**
- Implementation
- Data conversion and loading
- Testing  
**Add WeChat powcoder**
- Operational maintenance

# Stages of the Database System Development Lifecycle



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Database Planning (1 of 2)

- Management activities that allow stages of database system development lifecycle to be realized as efficiently and effectively as possible
- Must be integrated with overall IS strategy of the organization.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Database Planning – Mission Statement

- **Mission statement** for the database project defines major aims of database application.
- Those driving database project normally define the mission statement.  
<https://powcoder.com>
- Mission statement helps clarify purpose of the database project and provides clearer path towards the efficient and effective creation of required database system.

# Database Planning – Mission Objectives

- Once mission statement is defined, **mission objectives** are defined.
- Each objective should identify a particular task that the database must support.  
<https://powcoder.com>
- May be accompanied by some additional information that specifies the work to be done, the resources with which to do it, and the money to pay for it all.

# Database Planning (2 of 2)

- Database planning should also include development of standards that govern:
  - how data will be collected,
  - how the format should be specified,
  - what necessary documentation will be needed,
  - how design and implementation should proceed.

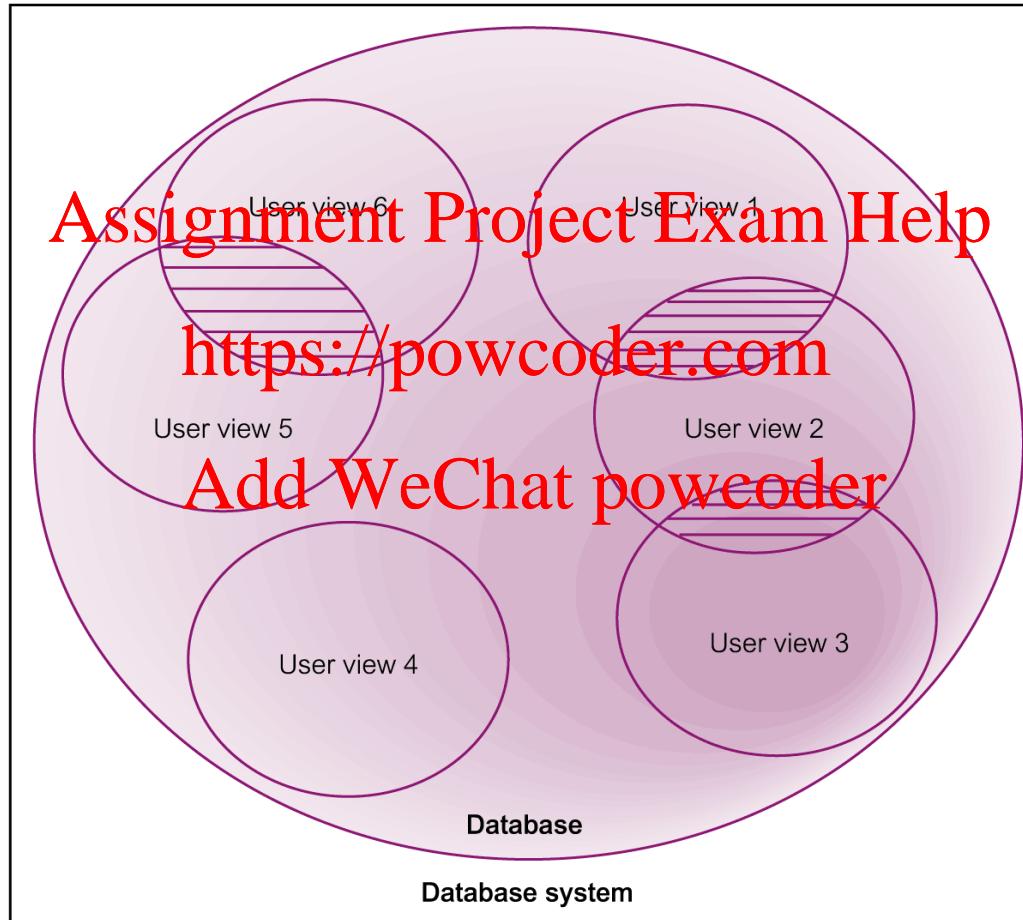
# System Definition (1 of 2)

- Describes scope and boundaries of database system and the major user views.
- User view defines what is required of a database system from perspective of:  
– a particular job role (such as Manager or Supervisor)  
or  
– enterprise application area (such as marketing, personnel, or stock control).

# System Definition (2 of 2)

- Database application may have one or more user views.
- Identifying user views helps ensure that no major users of the database are forgotten when developing requirements for new system.  
<https://powcoder.com>
- User views also help in development of complex database system allowing requirements to be broken down into manageable pieces.

# Representation of a Database System with Multiple User Views



# Requirements Collection and Analysis (1 of 4)

- Process of collecting and analyzing information about the part of organization to be supported by the database system, and using this information to identify users' requirements of new system.

<https://powcoder.com>

Add WeChat powcoder

# Requirements Collection and Analysis (2 of 4)

- Information is gathered for each major user view including:
  - a description of data used or generated
  - details of how data is to be used/generated;
  - any additional requirements for new database system.
- Information is analyzed to identify requirements to be included in new database system. Described in the requirements specification.

# Requirements Collection and Analysis (3 of 4)

- Another important activity is deciding how to manage the requirements for a database system with multiple user views. [Assignment](#) [Project](#) [Exam](#) [Help](#)
- Three main approaches:  
– centralized approach;  
– view integration approach;  
– combination of both approaches.

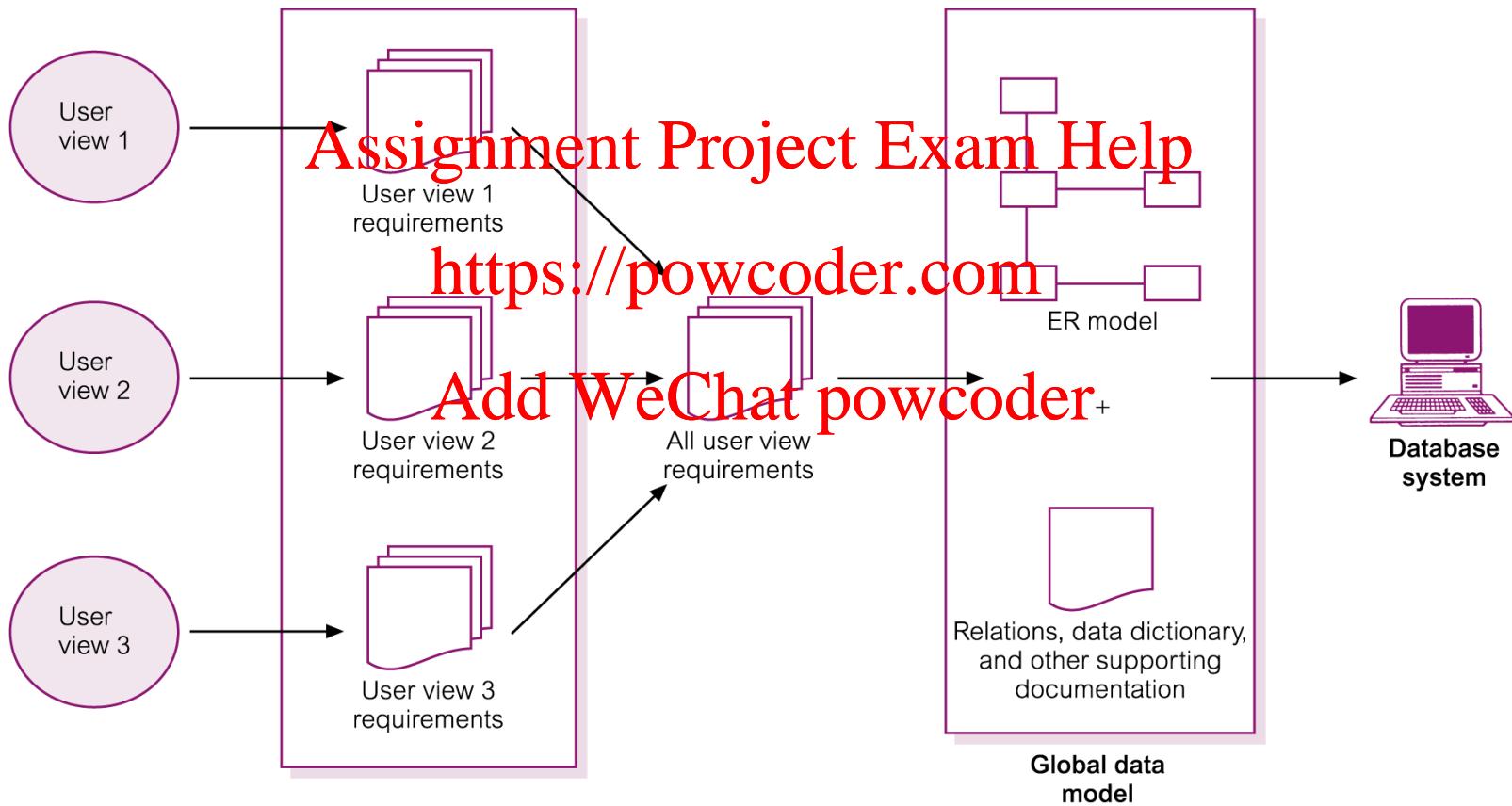
# Requirements Collection and Analysis (4 of 4)

- **Centralized approach**

- Requirements for each user view are merged into a single set of requirements.
- A data model is created representing all user views during the database design stage.

Add WeChat powcoder  
<https://powcoder.com>

# Centralized Approach to Managing Multiple User Views



# Requirements Collection and Analysis (1 of 3)

- **View integration approach**
  - Requirements for each user view remain as separate lists. [Assignment](#) [Project](#) [Exam](#) [Help](#)
  - Data models representing each user view are created and then merged later during the database design stage. <https://powcoder.com> [Add WeChat](#) [powcoder](#)

# Requirements Collection and Analysis (2 of 3)

- Data model representing single user view (or a subset of all user views) is called a **local data model**.
- Each model includes diagrams and documentation describing requirements for one or more but not all user views of database.

Add WeChat powcoder

# Requirements Collection and Analysis (3 of 3)

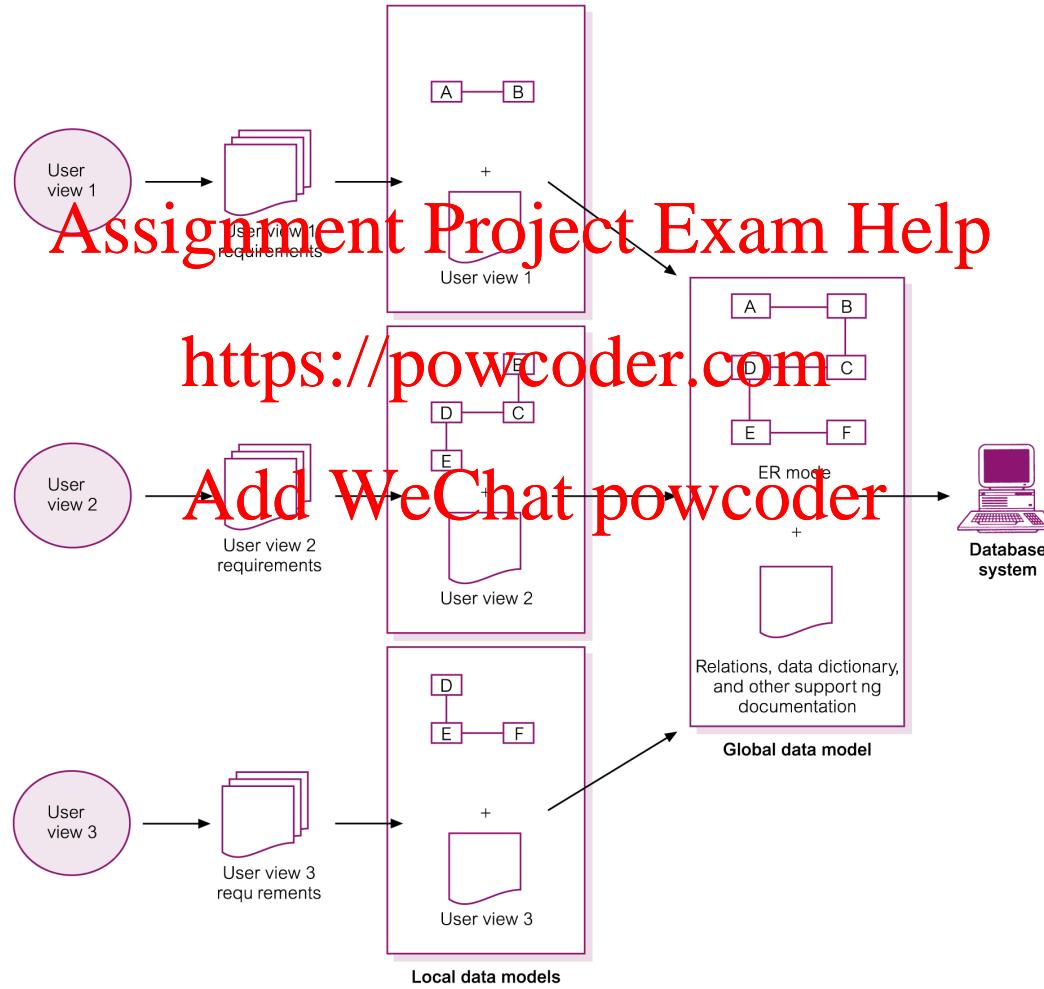
- Local data models are then merged at a later stage during database design to produce a **global data model**, which represents all user views for the database.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# View Integration Approach to Managing Multiple User Views



# Database Design (1 of 4)

- Process of creating a design for a database that will support the enterprise's mission statement and mission objectives for the required database system

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

# Database Design (2 of 4)

- Main approaches include:

- Top-down
- Bottom-up
- Inside-out
- Mixed

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Database Design (3 of 4)

- Main purposes of data modeling include:
  - to assist in understanding the meaning (semantics) of the data
  - to facilitate communication about the information requirements.
- Building data model requires answering questions about entities, relationships, and attributes.

# Database Design (4 of 4)

- A data model ensures we understand:
  - each user's perspective of the data;
  - nature of the data itself, independent of its physical representations;
  - use of data across user views.

Add WeChat powcoder

# Criteria to Produce an Optimal Data Model

<b>Structural validity</b>	Consistency with the way the enterprise defines and organizes information.
<b>Simplicity</b>	Ease of understanding by IS professionals and nontechnical users.
<b>Expressibility</b>	Ability to distinguish between different data, relationships between data, and constraints.
<b>Nonredundancy</b>	Exclusion of extraneous information; in particular, the representation of any one piece of information exactly once.
<b>Shareability</b>	Not specific to any particular application or technology and thereby usable by many.
<b>Extensibility</b>	Ability to evolve to support new requirements with minimal effect on existing users.
<b>Integrity</b>	Consistency with the way the enterprise uses and manages information.
<b>Diagrammatic representation</b>	Ability to represent a model using an easily understood diagrammatic notation.

# Database Design

- Three phases of database design:
  - Conceptual database design
  - Logical database design
  - Physical database design

Add WeChat powcoder

# Conceptual Database Design

- Process of constructing a model of the data used in an enterprise, independent of **all** physical considerations.
- Data model is built using the information in users' requirements specification.  
<https://powcoder.com>
- Conceptual data model is source of information for logical design phase.  
[Add WeChat powcoder](#)

# Logical Database Design

- Process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), **Assignment Project Exam Help** and other physical considerations.

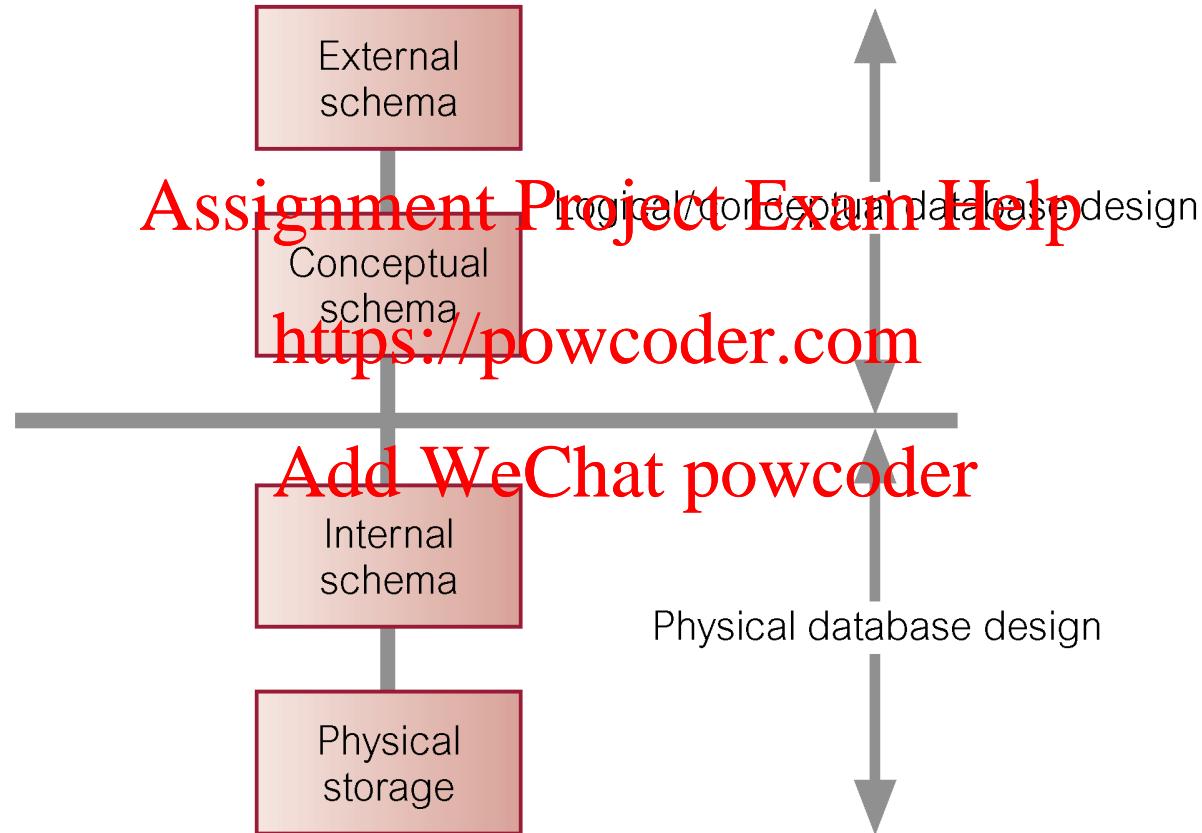
<https://powcoder.com>

- Conceptual data model is refined and mapped on to a logical data model **Add WeChat powcoder**

# Physical Database Design

- Process of producing a description of the database implementation on secondary storage.
- Describes base relations, file organizations, and indexes used to achieve efficient access to data. Also describes any associated integrity constraints and security measures.
- Tailored to a specific DBMS system.

# Three-Level ANSI-SPARC Architecture and Phases of Database Design



# DBMS Selection

- Selection of an appropriate DBMS to support the database system.
- Undertaken at any time prior to logical design provided sufficient information is available regarding system requirements.  
**Assignment Project Exam Help**  
**<https://powcoder.com>**
- Main steps to selecting a DBMS:
  - define Terms of Reference of study;
  - shortlist two or three products;
  - evaluate products;
  - recommend selection and produce report.

# DBMS Evaluation Features (1 of 4)

Data Definition	Physical Definition
Primary key enforcement	File structures available
Foreign key specification	File structure maintenance
Data types available	Ease of reorganization
Data type extensibility	Indexing
Domain specification	Variable length fields/records
Ease of restructuring	Data compression
Integrity controls	Encryption routines
View mechanism	Memory requirements
Data dictionary	Storage requirements
Data independence	
Underlying data model	
Schema evolution	

# DBMS Evaluation Features (2 of 4)

Accessibility	Transaction Handling
Query language: SQL2/SQL:2011/ODMG compliant	Backup and recovery routines Checkpointing facility
Interfacing to 3GLs	Assignment Project Exam Help Logging facility
Multi-user	Granularity of concurrency
Security	Deadlock resolution strategy
Access controls	Advanced transaction models
Authorization mechanism	Parallel query processing

# DBMS Evaluation Features (3 of 4)

Utilities	Development
Performance measuring	4GL/5GL tools
Tuning	<b>Assignment Project Exam Help</b> CASE tools
Load/unload facilities	Windows capabilities
User usage monitoring	Stored procedures, triggers, and rules
Database administration support	Web development tools
Other Features	
Upgradability	Interoperability with other DBMSs and other systems
Vendor stability	Web integration
User base	Replication utilities
Training and user support	Distributed capabilities

# DBMS Evaluation Features (4 of 4)

Other Features	
Documentation	Portability
Operating system required	Hardware required
Cost	Network support
Online help	Object-oriented capabilities
Standards used	Architecture (2- or 3-tier client/server)
Version management	Performance
Extensible query optimization	Transaction throughput
Scalability	Maximum number of concurrent users
Support for reporting and analytical tools	XML and Web services support

# Example - Evaluation of DBMS Product

DBMS: Sample product

Vendor: Sample vendor

## Physical Definition Group

Features	Comments	Rating	Weighting	Score
File structures available	Choice of 4	8	0.15	1.2
File structure maintenance	NOT self-regulating	6	0.2	1.2
Ease of reorganization		4	0.25	1.0
Indexing		6	0.15	0.9
Variable length fields/records		6	0.15	0.9
Data compression	Specify with file structure	7	0.05	0.35
Encryption routines	Choice of 2	4	0.05	0.2
Memory requirements		0	0.00	0
Storage requirements		0	0.00	0
Totals		41	1.0	<b>5.75</b>
Physical definition group		5.75	0.25	<b>1.44</b>

Add WeChat powcoder

<https://powcoder.com>

# Application Design

- Design of user interface and application programs that use and process the database.
- Database design and application design are parallel activities.  
<https://powcoder.com>
- Includes two important activities:
  - transaction design;
  - user interface design.

# Application Design - Transactions (1 of 2)

- An action, or series of actions, carried out by a single user or application program, which accesses or changes content of the database.
- Should define and document the high-level characteristics of the transactions required.

Add WeChat powcoder

# Application Design - Transactions (2 of 2)

- Important characteristics of transactions:
  - data to be used by the transaction;
  - functional characteristics of the transaction;
  - output of the transaction;  
<https://powcoder.com>
  - importance to the users;
  - expected rate of usage.
- Three main types of transactions: retrieval, update, and mixed.

# Prototyping

- Building working model of a database system.
- Purpose
  - to identify features of a system that work well, or are inadequate; <https://powcoder.com>
  - to suggest improvements or even new features;
  - to clarify the users' requirements;
  - to evaluate feasibility of a particular system design.

# Implementation

- Physical realization of the database and application designs.
  - Use DDA to create database schemas and empty database files.
  - Use DDL to create any specified user views.
  - Use 3GL or 4GL to create the application programs. This will include the database transactions implemented using the DML, possibly embedded in a host programming language.

# Data Conversion and Loading

- Transferring any existing data into new database and converting any existing applications to run on new database. **Assignment Project Exam Help**
- Only required when new database system is replacing an old system.
  - DBMS normally has utility that loads existing files into new database.
- May be possible to convert and use application programs from old system for use by new system.

# Testing (1 of 2)

- Process of running the database system with intent of finding errors.
- Use carefully planned test strategies and realistic data.
- Testing cannot show absence of faults; it can show only that software faults are present.  
<https://powcoder.com>  
Add WeChat powcoder
- Demonstrates that database and application programs appear to be working according to requirements.

## Testing (2 of 2)

- Should also test usability of system.
- Evaluation conducted against a usability specification.  
**Assignment Project Exam Help**
- Examples of criteria include:
  - Learnability; <https://powcoder.com>
  - Performance; [Add WeChat powcoder](#)
  - Robustness;
  - Recoverability;
  - Adaptability.

# Operational Maintenance

- Process of monitoring and maintaining database system following installation.
- Monitoring performance of system.
  - if performance falls, may require tuning or reorganization of the database.
- Maintaining and upgrading database application (when required).
- Incorporating new requirements into database application.

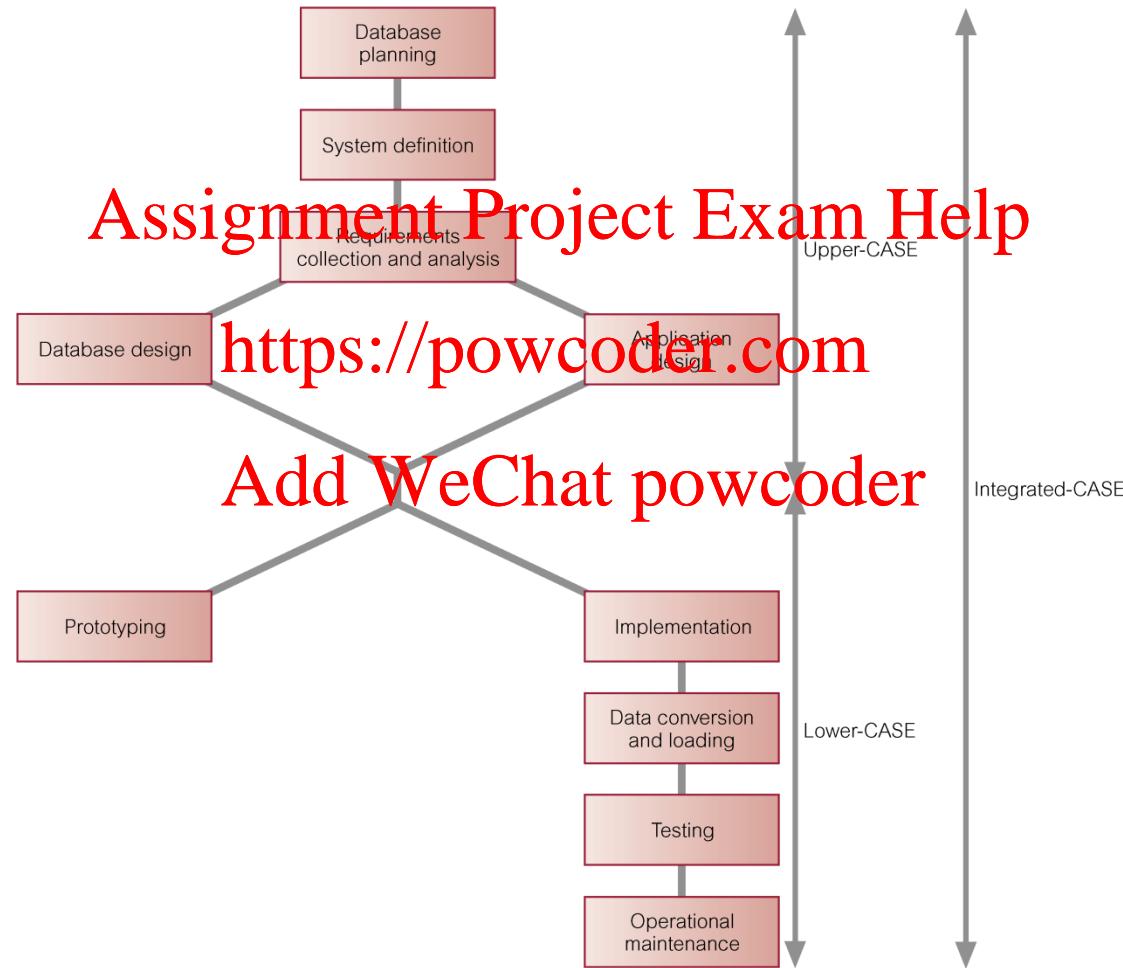
# CASE Tools (1 of 2)

- Support provided by CASE tools include:
  - data dictionary to store information about database system's **Assignment Project Exam Help**
  - design tools to support data analysis;
  - tools to permit development of corporate data model, and conceptual **Add WeChat powcoder**;
  - tools to enable prototyping of applications.

## CASE Tools (2 of 2)

- Provide following benefits:
  - Standards;
  - Integration;
  - Support for standard methods;  
<https://powcoder.com>
  - Consistency;
  - Automation.

# CASE Tools and Database System Development Lifecycle



# Data Administration and Database Administration

- The Data Administrator (DA) and Database Administrator (DBA) are responsible for managing and controlling the corporate data and corporate database, respectively.
- DA is more concerned with early stages of database system development lifecycle and DBA is more concerned with later stages.

# Data Administration

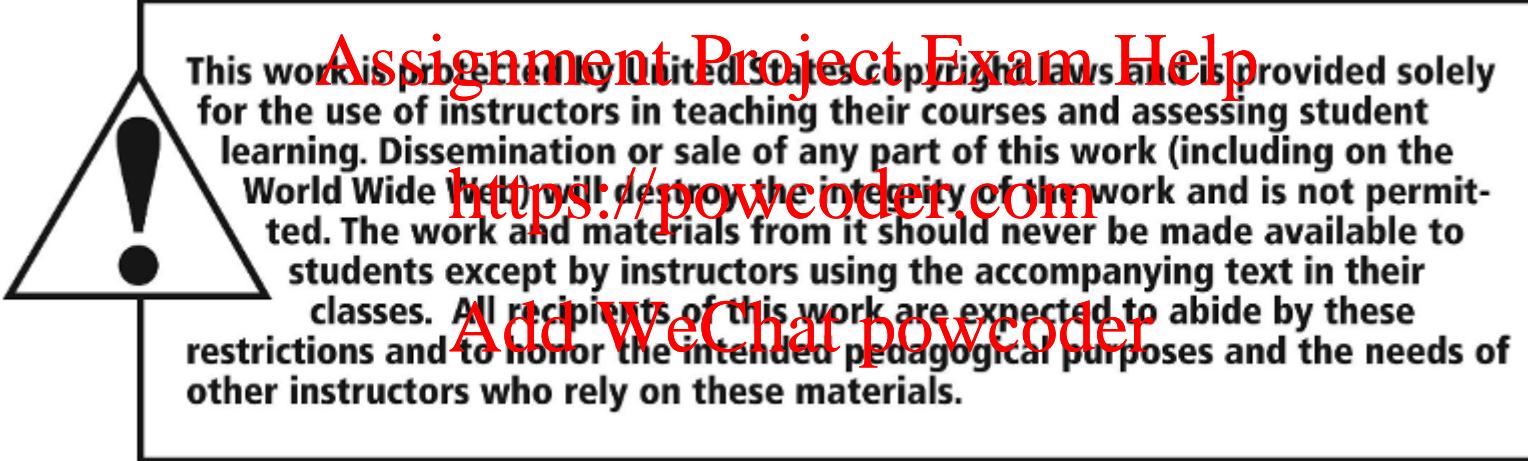
- Management of data resource including:
  - database planning,
  - development and maintenance of standards, policies and procedures, and conceptual and logical database design.

Add WeChat powcoder

# Database Administration

- Management of physical realization of a database system including:
  - physical database design and implementation, [Assignment Project Example Help](https://powcoder.com), <https://powcoder.com>
  - setting security and integrity controls,
  - monitoring system performance, and reorganizing the database. [Add WeChat powcoder](https://powcoder.com)

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 11

<https://powcoder.com>  
Database Analysis and the  
**DreamHome** Case Study  
Add WeChat powcoder

# Learning Objectives (1 of 3)

- 11.1** When fact-finding techniques are used in the database application lifecycle.
- 11.2** The types of facts collected in each stage of the database application lifecycle.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- 11.3** The types of documentation produced in each stage of the database application lifecycle.  
[Add WeChat powcoder](#)

# Learning Objectives (2 of 3)

**11.4** The most commonly used fact-finding techniques.

**11.5** How to use each fact-finding technique and the advantages and disadvantages of each.  
*Assignment Project Exam Help*

**11.6** About a property rental company called **DreamHome**.  
*<https://powcoder.com>*

Add WeChat powcoder

# Learning Objectives (3 of 3)

**11.7** How to apply fact-finding techniques to the early stages of the database application lifecycle.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Fact-Finding Techniques (1 of 2)

- It is critical to capture the necessary facts to build the required database application.
- These facts are captured using fact-finding techniques.
- The formal process of using techniques such as interviews and questionnaires to collect facts about systems, requirements, and preferences.

# When Are Fact-Finding Techniques Used?

- Fact-finding used throughout the database application lifecycle. Crucial to the early stages including database planning, system definition, and requirements collection and analysis stages.
- Enables developer to learn about the terminology, problems, opportunities, constraints, requirements, and priorities of the organization and the users of the system.

<https://powcoder.com>

Add WeChat powcoder

# Examples of Data Captured and Documentation Produced During the Database Application Lifecycle (1 of 3)

Stage of Database System Development Lifecycle	Examples of Data Captured	Examples of Documentation Produced
Database planning	Aims and objectives of database project	Mission statement and objectives of database system
System definition	Description of major user views (includes job roles or business application areas)	Definition of scope and boundary of database system; definition of user views to be supported
Requirements collection and analysis	Requirements for user views; systems specifications, including performance and security requirements	Users' and system requirements specifications
Database design	Users' responses to checking the conceptual/logical database design; functionality provided by target DBMS	Conceptual/logical database design (includes ER model(s), data dictionary, and relational schema); physical database Design

# Examples of Data Captured and Documentation Produced During the Database Application Lifecycle (2 of 3)

Stage of Database System Development Lifecycle	Examples of Data Captured	Examples of Documentation Produced
Application design	Users' responses to checking interface design	Application design (includes description of programs and user interface)
DBMS selection	Functionality provided by target DBMS	DBMS evaluation and recommendations
Prototyping	Users' responses to prototype	Modified users' requirements and systems specifications
Implementation	Functionality provided by target DBMS	
Data conversion and loading	Format of current data; data import capabilities of target DBMS	

# Examples of Data Captured and Documentation Produced During the Database Application Lifecycle (3 of 3)

Stage of Database System Development Lifecycle	Examples of Data Captured	Examples of Documentation Produced
Testing	Test results	Testing strategies used; analysis of test results
Operational maintenance	Performance testing results; new or changing user and system requirements	User manual; analysis of performance results; modified users' requirements and systems specifications

# Fact-Finding Techniques (2 of 2)

- A database developer normally uses several fact-finding techniques during a single database project including:
  - examining documentation
  - interviewing
  - observing the organization in operation
  - research
  - questionnaires

# Examining Documentation

- Can be useful
  - to gain some insight as to how the need for a database **Assignment Project Exam Help**
  - to identify the part of the organization associated with the problem. **https://powcoder.com**
  - To understand **Add WeChat post** **powcoder**

# Examples of Types of Documentation That Should Be Examined

Purpose of Documentation	Examples of Useful Sources
Describes problem and need for database	Internal memos, emails, and minutes of meetings Employee complaints and documents that describe the problem Performance reviews/reports
Describes the part of the enterprise affected by Problem	Organizational chart, mission statement, and strategic plan of the enterprise Objectives for the part of the enterprise being studied Task/job descriptions Samples of completed manual forms and reports Samples of completed computerized forms and reports
Describes current system	Various types of flowcharts and diagrams Data dictionary Database system design Program documentation User/training manuals

# Interviewing (1 of 2)

- Most commonly used, and normally most useful, fact-finding technique. Enables collection of information from individuals face-to-face.
- Objectives include finding out facts, verifying facts, clarifying facts, generating enthusiasm, getting the end-user involved, identifying requirements, and gathering ideas and opinions.

# Advantages and Disadvantages of Interviewing

## Advantages

Allows interviewee to respond freely and openly to questions

Allows interviewee to feel part of project

Allows interviewer to follow up on interesting comments made by interviewee

Allows interviewer to adapt or reword questions during interview

Allows interviewer to observe interviewee's body language

## Disadvantages

Very time-consuming and costly, and therefore may be impractical

Success is dependent on communication skills of interviewer

Success can be dependent on willingness of interviewees to participate in interviews

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## Interviewing (2 of 2)

- There are two types of interviews unstructured and structured.
- Open-ended questions allow the interviewee to respond in any way that seems appropriate.  
<https://powcoder.com>
- Closed-ended questions restrict answers to either specific choices or short, direct responses.

# Observing the Organization in Operation

- An effective technique for understanding a system.
- Possible to either participate in, or watch, a person perform activities to learn about the system.
- Useful when validity of data collected is in question or when the complexity of certain aspects of the system prevents a clear explanation by the end-users.

# Advantages and Disadvantages of Using Observation

Advantages	Disadvantages
Allows the validity of facts and data to be Checked	People may knowingly or unknowingly perform differently when being observed
Observer can see exactly what is being done	May miss observing tasks involving different levels of difficulty or volume normally experienced during that time period
Observer can also obtain data describing the physical environment of the task	Some tasks may not always be performed in the manner in which they are observed
Relatively inexpensive	May be impractical
Observer can do work measurements	

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Observer can also obtain data describing the physical environment of the task

Relatively inexpensive

Observer can do work measurements

# Research

- Useful to research the application and problem.
- Use computer trade journals, reference books, and the Internet (including user groups and bulletin boards).
- Provide information on how others have solved similar problems, plus whether or not software packages exist to solve or even partially solve the problem.

# Advantages and Disadvantages of Using Research

---

## Advantages

Can save time if solution already exists

Researcher can see how others have solved similar problems or met similar requirements

Keeps researcher up to date with current developments

## Disadvantages

Requires access to appropriate sources of information

May ultimately not help in solving problem because problem is not documented elsewhere

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Questionnaires

- Conduct surveys through questionnaires, which are special-purpose documents that allow facts to be gathered from a large number of people while maintaining some control over their responses.  
<https://powcoder.com>
- There are two types of questions, namely free-format and fixed-format.  
[Add WeChat powcoder](#)

# Advantages and Disadvantages of Using Questionnaires

Advantages	Disadvantages
People can complete and return questionnaires at their convenience	Number of respondents can be low, possibly only 5% to 10%
Relatively inexpensive way to gather data from a large number of people	Questionnaires may be returned incomplete
People more likely to provide the real facts as responses can be kept confidential	May not provide an opportunity to adapt or reword questions that have been misinterpreted
Responses can be tabulated and analyzed quickly	Cannot observe and analyze the respondent's body language

# Using Fact-Finding Techniques – A Worked Example (1 of 7)

*DreamHome*  
Staff Registration Form

Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder

Staff Number <input type="text" value="S001"/>	Branch Number <input type="text" value="5005"/>
Full Name <input type="text" value="Susan Brand"/>	Branch Address <input type="text" value="103 Main St, Glasgow"/>
Sex <input type="text" value="F"/>	DOB <input type="text" value="3-Jun-70"/>
Position <input type="text" value="Manager"/>	Telephone Number(s) <input type="text" value="0141-339-2175 / 0141-339-4439"/>
Salary <input type="text" value="24000"/>	
Enter details where applicable	
Supervisor Name <input type="text" value=" "/>	Manager Start Date <input type="text" value="01-Jun-99"/>
	Manager Bonus <input type="text" value="2350"/>

# Using Fact-Finding Techniques – A Worked Example (2 of 7)

*DreamHome*  
Staff Listing

Assignment Project Exam Help  
Branch Number 5000 Branch Address 163 Main St, Glasgow  
Telephone Number(s) 0141-339-2178 / 0141-339-4555 G1 9QX

Add WeChat powcoder

Staff Number	Name	Position
SG5	Susan Brand	Manager
SG14	David Ford	Supervisor
SG37	Ann Beech	Assistant
SG112	Annet Longhorn	Supervisor
SG126	Chris Lawrence	Assistant
SG132	Sofie Walters	Assistant

Page 1

# Using Fact-Finding Techniques – A Worked Example (3 of 7)

DreamHome Property Registration Form	
<p>Property Number <u>PG16</u> Type <u>Flat</u> Room <u>4</u> Rent <u>450</u> Address <u>5 Novar Drive,</u> <u>Glasgow G12 9AX</u></p> <p style="color: red; font-size: 2em; margin-top: 10px;">Assignment Project Exam Help <a href="https://powcoder.com">https://powcoder.com</a> Add WeChat powcoder</p>	<p>Owner Number <u>C093</u> (If known) Person/Business Name <u>Tony Shaw</u> Address <u>12 Park Pl,</u> <u>Glasgow G4 0QR</u> Tel No <u>0141-225-7000</u></p>
<p>Enter details where applicable</p> <p>Type of business _____</p> <p>Contact Name _____</p>	
<p>Managed by staff <u>David Ford</u></p>	<p>Registered at branch <u>163 Main St, Glasgow</u></p>

# Using Fact-Finding Techniques – A Worked Example (4 of 7)

*DreamHome*  
Client Registration Form

**Assignment Project Exam Help**

<p>Client Number <u>CR74</u> (Enter if known) <u><a href="https://powcoder.com">https://powcoder.com</a></u></p> <p>Full Name <u>Mike Ritson</u></p>	<p>Branch Number <u>B003</u> Branch Address <u>163 Main St, Glasgow</u></p> <p>Registered By <u>Ann Beech</u></p> <p>Date Registered <u>16-Nov-11</u></p>
<p>Enter property requirements</p> <p>Type <u>Flat</u></p> <p>Max Rent <u>750</u></p>	

# Using Fact-Finding Techniques – A Worked Example (5 of 7)

*DreamHome*  
Property Listing for Week beginning 01/06/13

If you are interested in viewing or renting any of the properties in this list, please contact the branch office as soon as possible.

**Assignment Project Exam Help**  
<https://powcoder.com>

**Add WeChat powcoder**

Property No	Address	Type	Rooms	Rent
PG4	6 Lawrence St, Glasgow	Flat	3	350
PG36	2 Manor Rd, Glasgow	Flat	3	375
PG21	18 Dale Road, Glasgow	House	5	600
PG16	5 Novar Drive, Glasgow	Flat	4	450
PG77	100A Apple Lane, Glasgow	House	6	560
PG81	781 Greentree Dr, Glasgow	Flat	4	440

Page 1

# Using Fact-Finding Techniques – A Worked Example (6 of 7)

DreamHome Property Viewing Report			
Property Number	PG4		
Type	Flat		
Rent	£350		
Assignment Project Exam Help <a href="https://powcoder.com">https://powcoder.com</a>			
Add WeChat powcoder			
Client No	Name	Date	Comments
CR76	John Kay	20/04/13	Too remote.
CR56	Aline Stewart	26/05/13	
CR74	Mike Ritchie	11/11/13	
CR62	Mary Tregear	11/11/13	OK, but needs redecoration throughout.

# Using Fact-Finding Techniques – A Worked Example (7 of 7)

<p><i>DreamHome Lease Number 00345810</i></p>	
<p>Client Number <u>00345810</u> (Enter if known)</p> <p>Full Name <u>Mike Kitchie</u> (Please print)</p> <p>Client Signature <u>Add WeChat powcoder</u></p>	<p>Property Number <u>00345</u></p> <p>Property Address <u>5 Novar Dr, Glasgow</u></p>
<p>Enter payment details</p> <p>Monthly Rent <u>450</u></p> <p>Payment Method <u>Cheque</u></p> <p>Deposit Paid (Y or N) <u>Yes</u></p>	<p>Rent Start <u>01/06/12</u></p> <p>Rent Finish <u>31/05/13</u></p> <p>Duration <u>1 year</u></p>

# Mission Statement for DreamHome Database System

“The purpose of the **DreamHome** database system is to maintain the data that is used and generated to support the property rentals business for our clients and property owners and to facilitate the cooperation and sharing of information between branches.”

Add WeChat powcoder

# Mission Objectives for DreamHome Database System

To maintain (enter, update, and delete) data on branches.  
To maintain (enter, update, and delete) data on staff.  
To maintain (enter, update, and delete) data on properties for rent.  
To maintain (enter, update, and delete) data on property owners.  
To maintain (enter, update, and delete) data on clients.  
To maintain (enter, update, and delete) data on property viewings.  
To maintain (enter, update, and delete) data on leases.  
To maintain (enter, update, and delete) data on newspaper adverts.

**Assignment Project Exam Help**

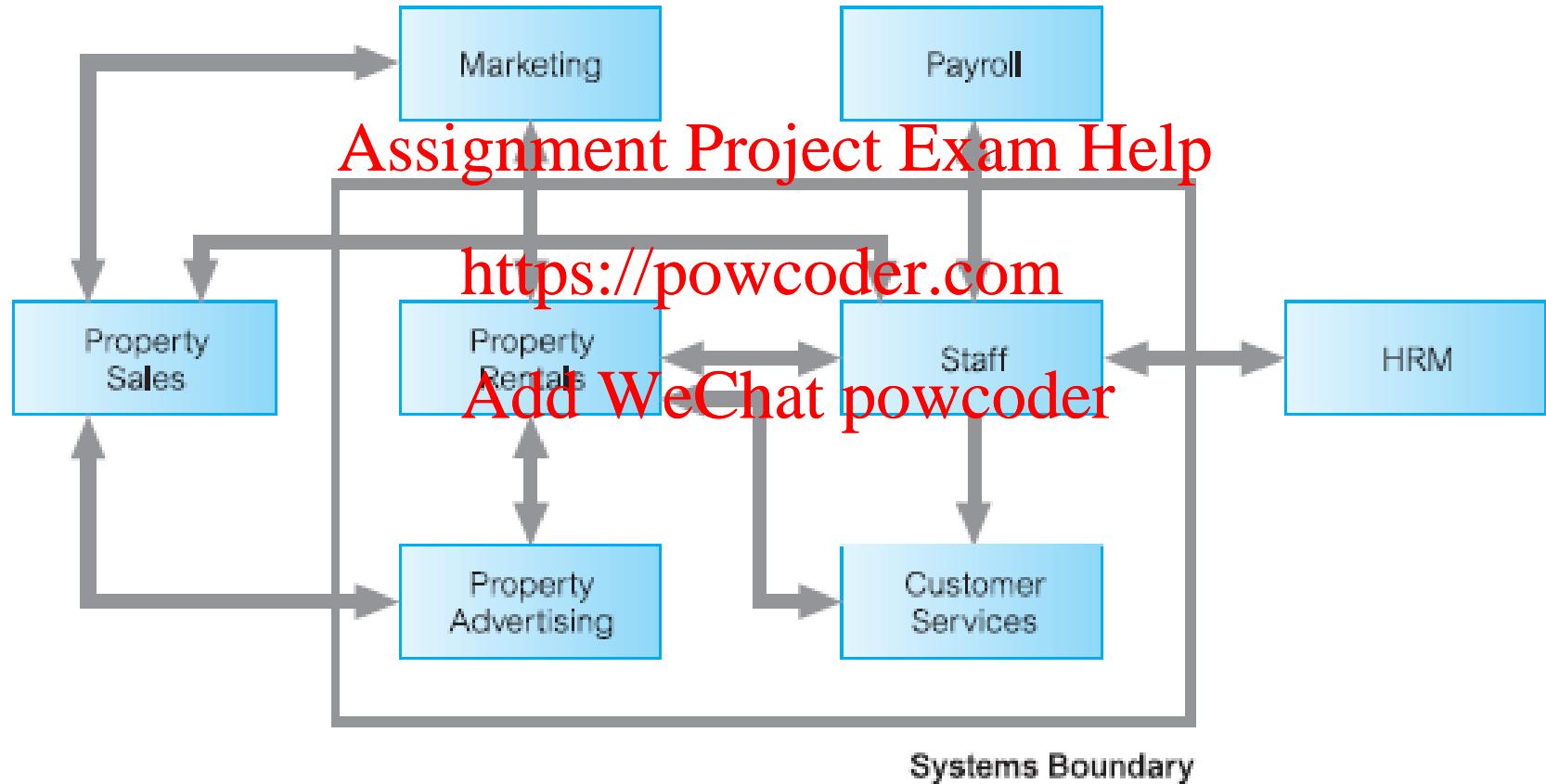
To perform searches on branches.  
To perform searches on staff.  
To perform searches on properties for rent.  
To perform searches on property owners.  
To perform searches on clients.  
To perform searches on property viewings.  
To perform searches on leases.  
To perform searches on newspaper adverts.

**Add WeChat powcoder**

To track the status of property for rent.  
To track the status of clients wishing to rent.  
To track the status of leases.

To report on branches.  
To report on staff.  
To report on properties for rent.  
To report on property owners.  
To report on clients.  
To report on property viewings.  
To report on leases.  
To report on newspaper adverts.

# System Boundary for DreamHome Database System



# Major User Views for DreamHome Database System

Data	Access Type	Director	Manager	Supervisor	Assistant
All Branches	Maintain				
	Query	X	X		
	Report	X	X		
	Maintain		X		
	Query		X		
	Report		X		
Single Branch	Maintain				
	Query	X	X		
	Report	X	X		
	Maintain		X		
	Query		X		
	Report		X		
All Staff	Maintain				
	Query	X	X		
	Report	X	X		
	Maintain		X		
	Query		X	X	
	Report		X	X	
Branch Staff	Maintain				
	Query	X	X		
	Report	X	X		
	Maintain		X		
	Query		X	X	
	Report		X	X	
All Property	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
Branch Property	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
All Owners	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
Branch Owners	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
All Clients	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
Branch Clients	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
All Viewings	Maintain				
	Query				
	Report				
	Maintain			X	X
	Query			X	X
	Report			X	X
Branch Viewings	Maintain				
	Query				
	Report				
	Maintain			X	X
	Query			X	X
	Report			X	X
All Leases	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
Branch Leases	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X	X	
	Query		X	X	X
	Report		X	X	X
All Newspapers	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X		
	Query		X		
	Report		X		
Branch Newspapers	Maintain				
	Query	X			
	Report	X	X		
	Maintain		X		
	Query		X		
	Report		X		

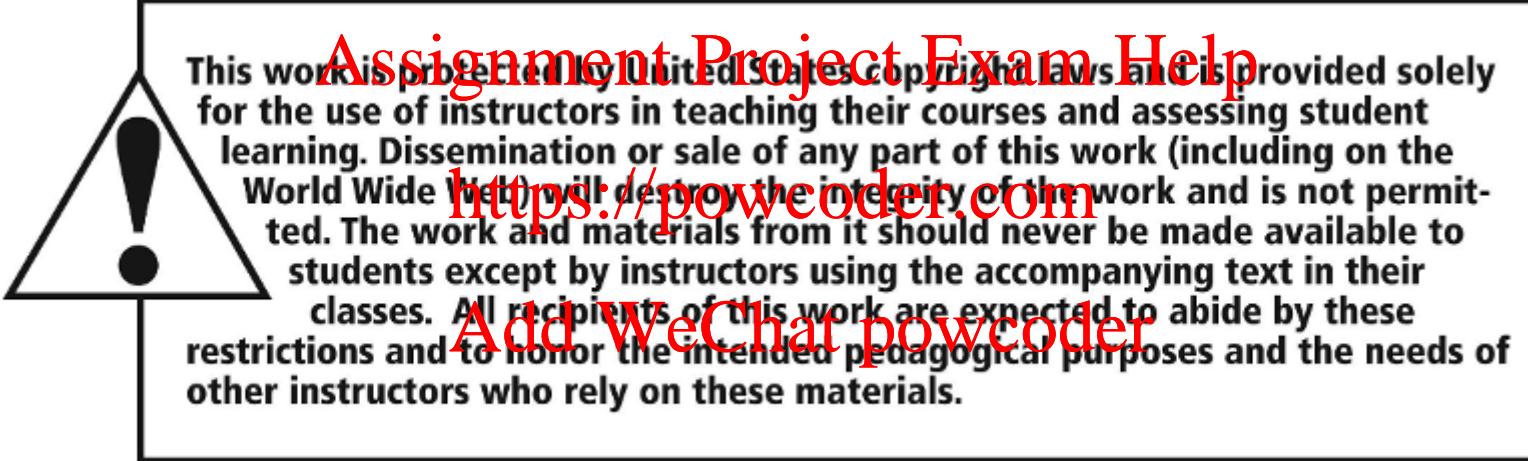
Assignment Project Exam Help

<https://powcoder.com>  
Add WeChat powcoder

# Cross-Reference of User Views with Main Types of Data Used by Each

	Director	Manager	Supervisor	Assistant
branch	X	X		
staff	X	X	X	
property for rent	X	X	X	X
owner	X	X	X	X
client	X	X	X	X
property viewing			X	X
lease	X	X	X	X
newspaper	X	X		

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 12

Entity-Relationship Modeling

Add WeChat powcoder

# Learning Objectives

**12.1** How to use Entity–Relationship (ER) modeling in database design.

**12.2** Basic concepts associated with ER model.

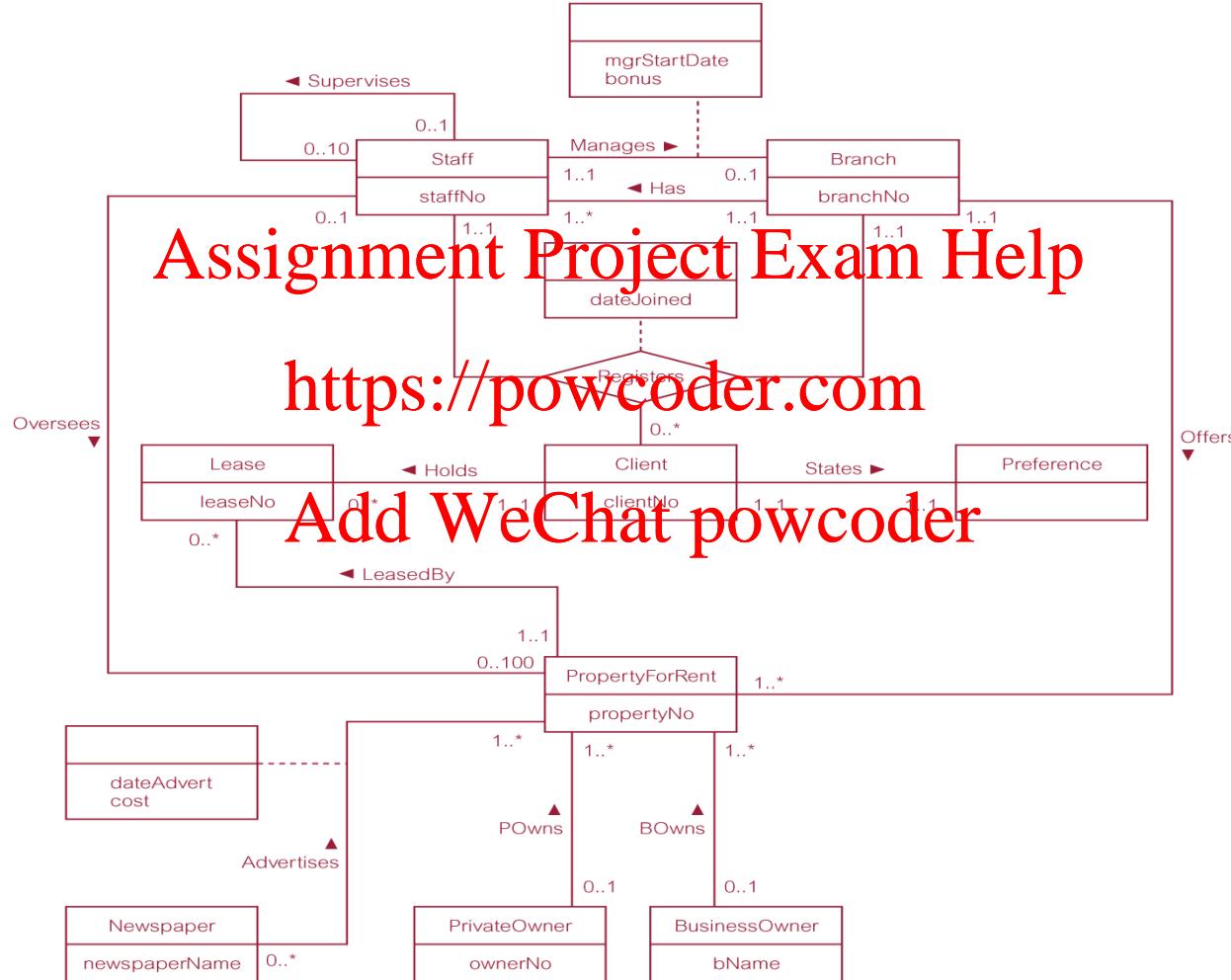
**12.3** Diagrammatic technique for displaying ER model using Unified Modeling Language (UML).

Add WeChat powcoder

**12.4** How to identify and resolve problems with ER models called connection traps.

**12.5** How to build an ER model from a requirements specification.

# ER Diagram of Branch User Views of DreamHome



# Concepts of the ER Model

- Entity types
- Relationship types
- Attributes

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Entity Type (1 of 2)

- Entity type
  - Group of objects with same properties, identified by enterprise assignment having independent existence.
- Entity occurrence <https://powcoder.com>
  - Uniquely identifiable object of an entity type.  
[Add WeChat powcoder](#)

# Examples of Entity Types

## Physical existence

Staff

Part

Property

Supplier

Customer

Product

Assignment Project Exam Help

<https://powcoder.com>

## Conceptual existence

Viewing

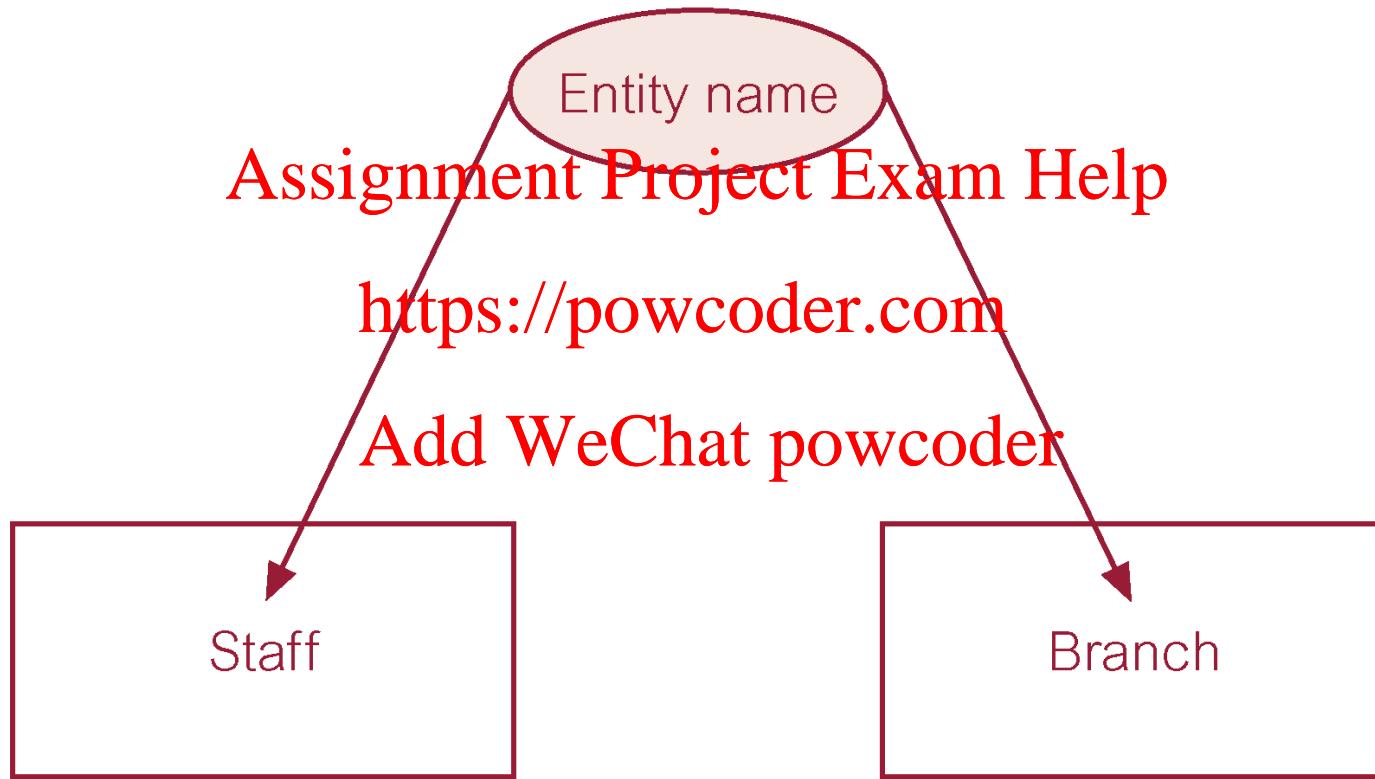
Sale

Inspection

Work experience

Add WeChat powcoder

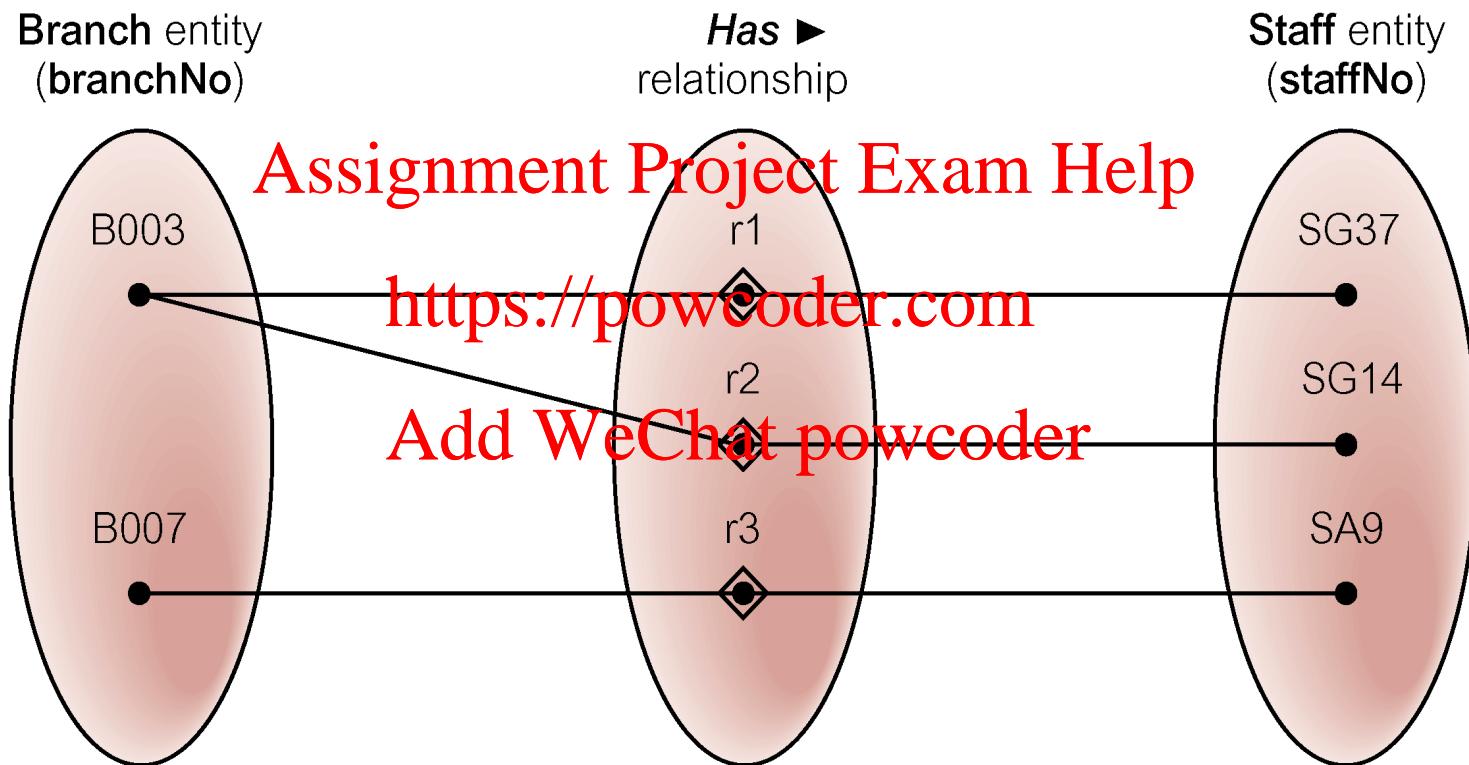
# ER Diagram of Staff and Branch Entity Types



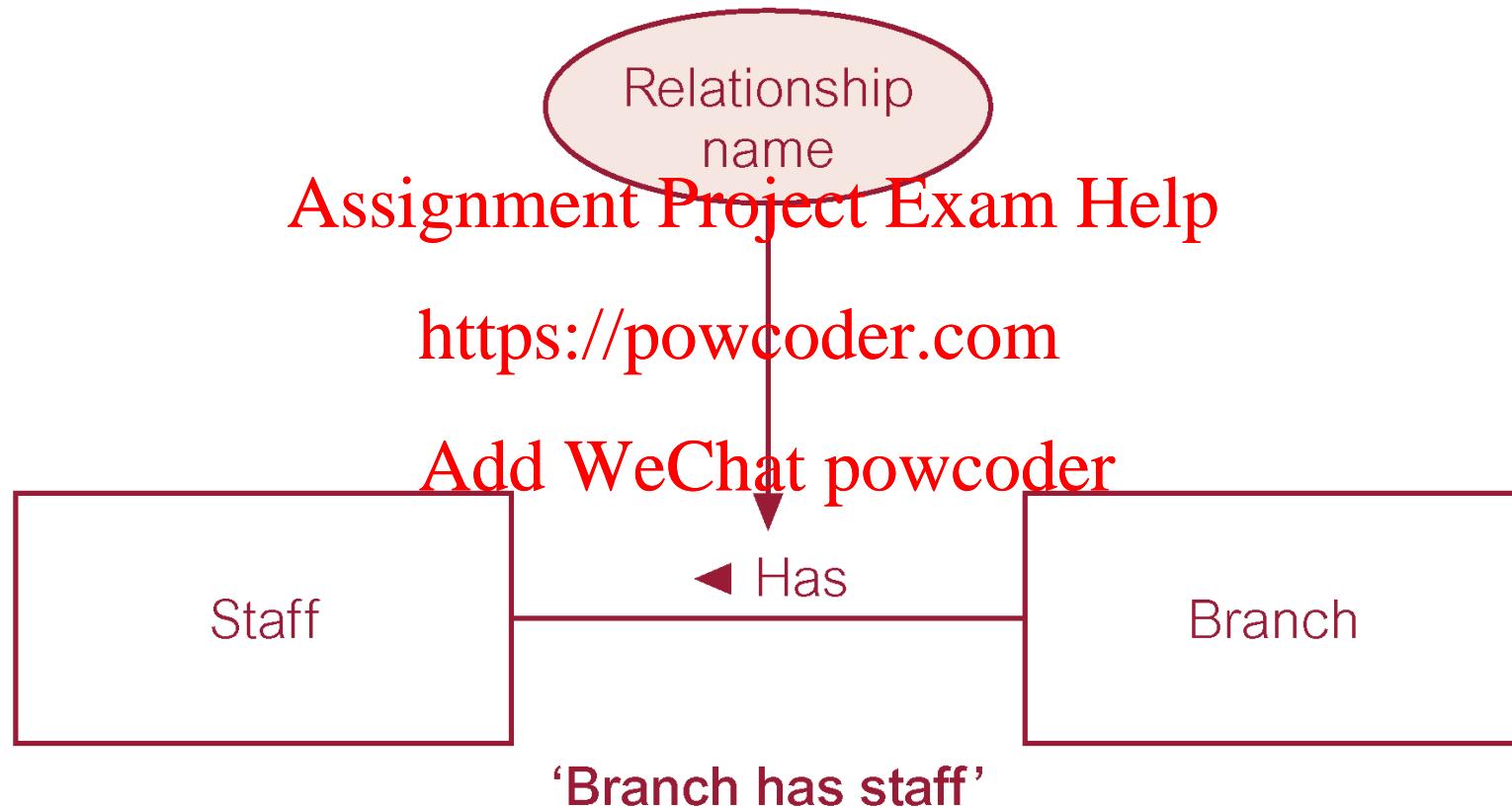
# Relationship Types

- Relationship type
  - Set of meaningful associations among entity types.
- Relationship occurrence
  - Uniquely identifiable association, which includes one occurrence from each participating entity type.

# Semantic Net of Has Relationship Type



# ER Diagram of Branch Has Staff Relationship



# Relationship Types (1 of 2)

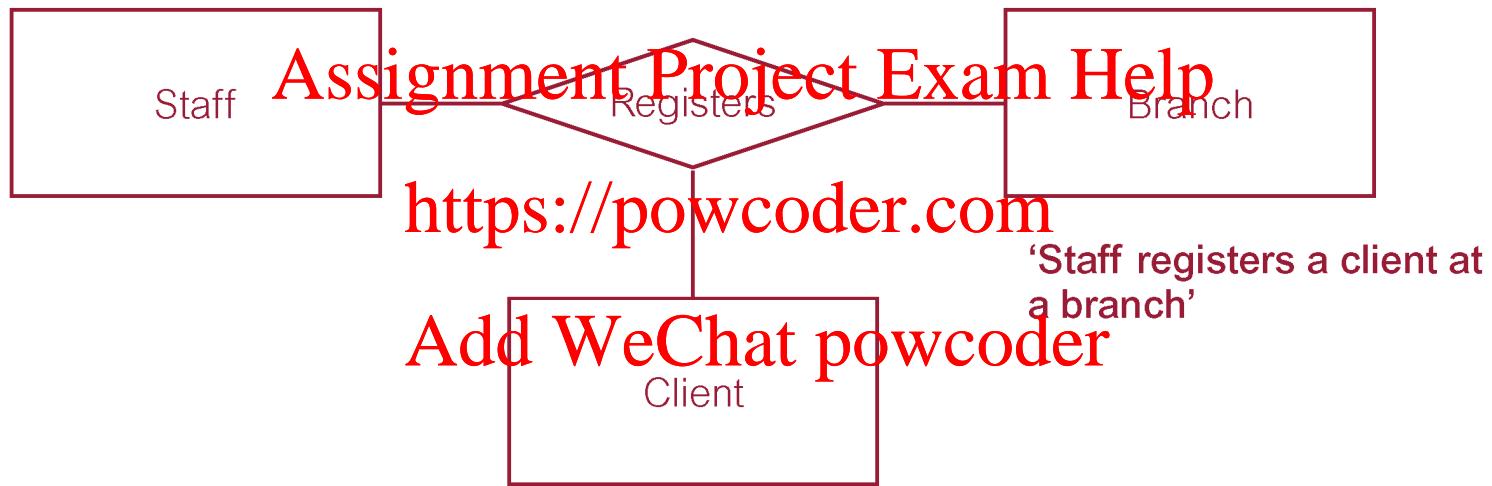
- Degree of a Relationship
  - Number of participating entities in relationship.
- Relationship of degree:
  - two is binary
  - three is ternary
  - four is quaternary.

# Binary Relationship Called POwNs

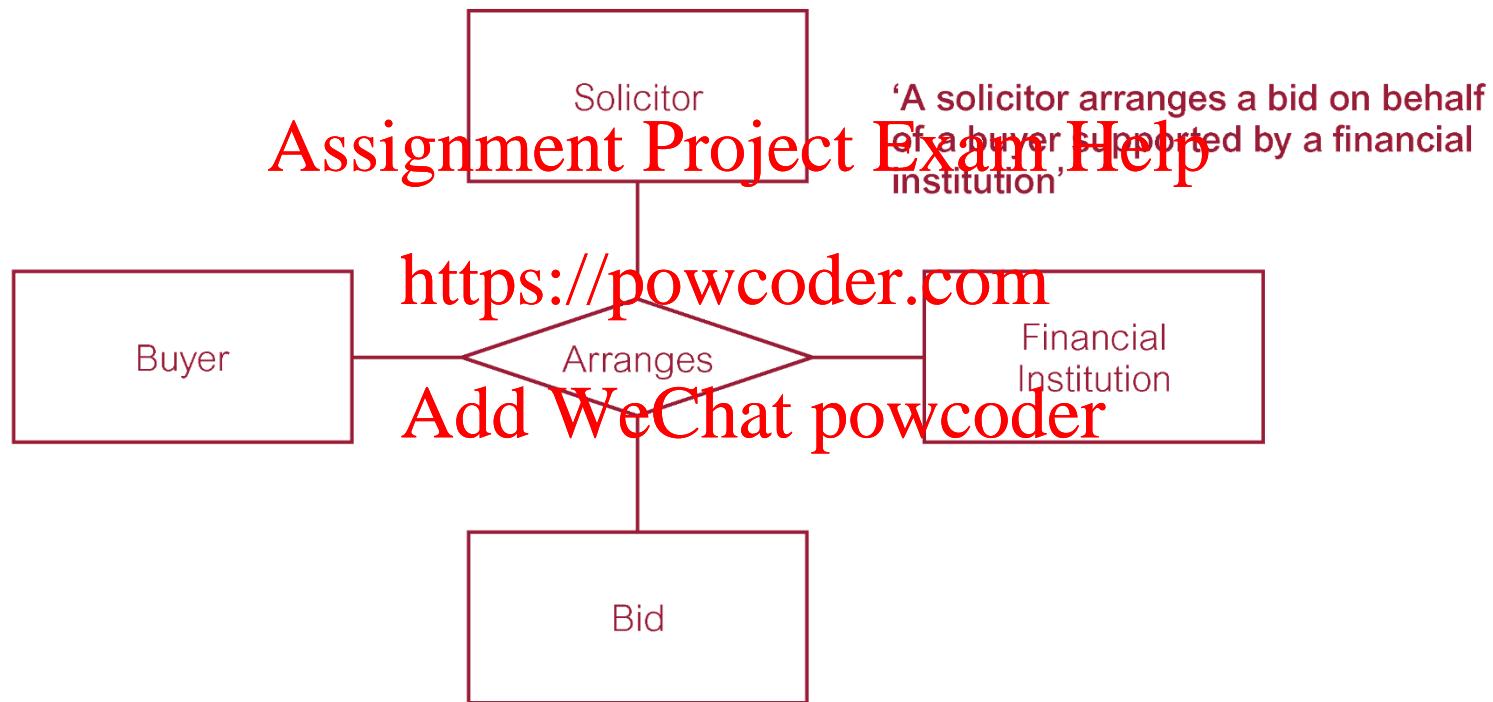
Assignment Project Exam Help



# Ternary Relationship Called Registers



# Quaternary Relationship Called Arranges

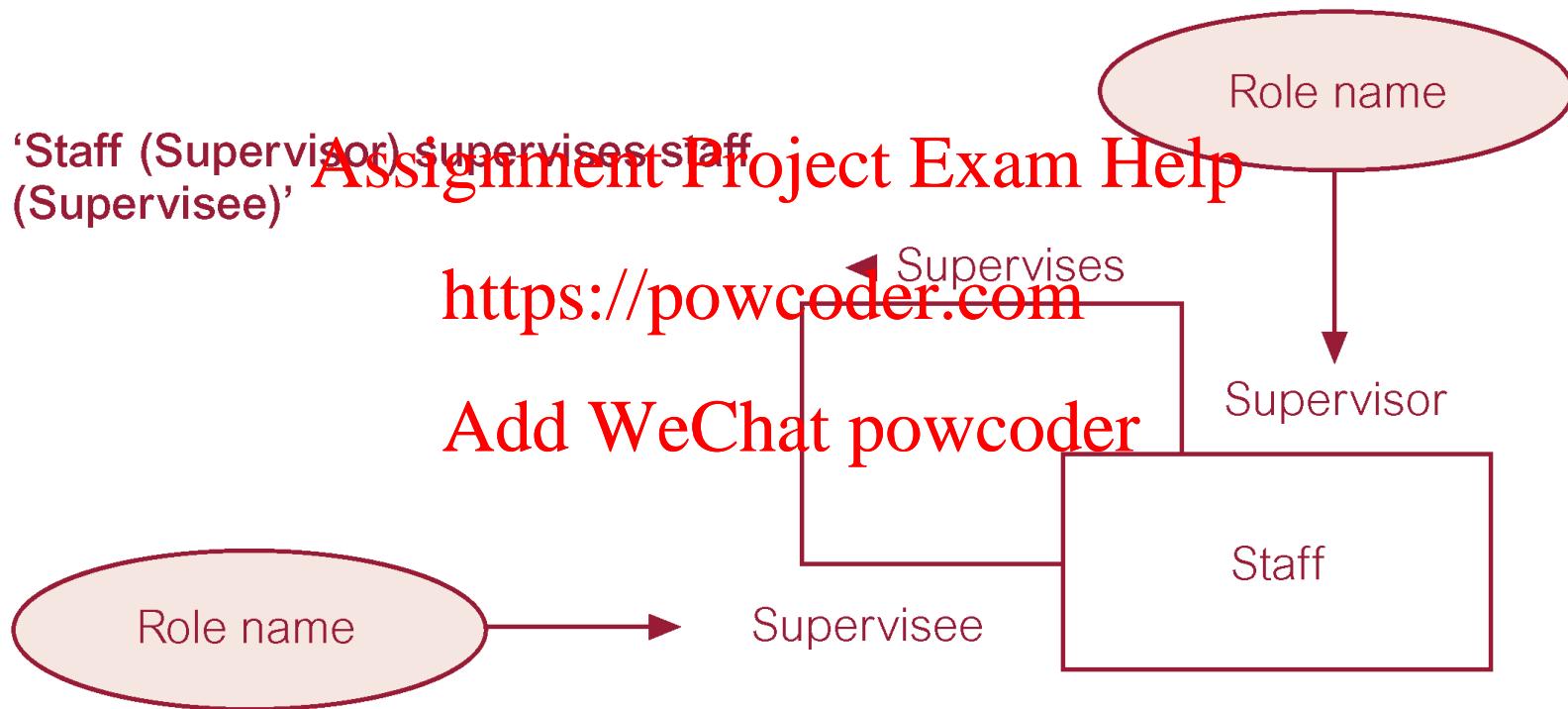


# Relationship Types (2 of 2)

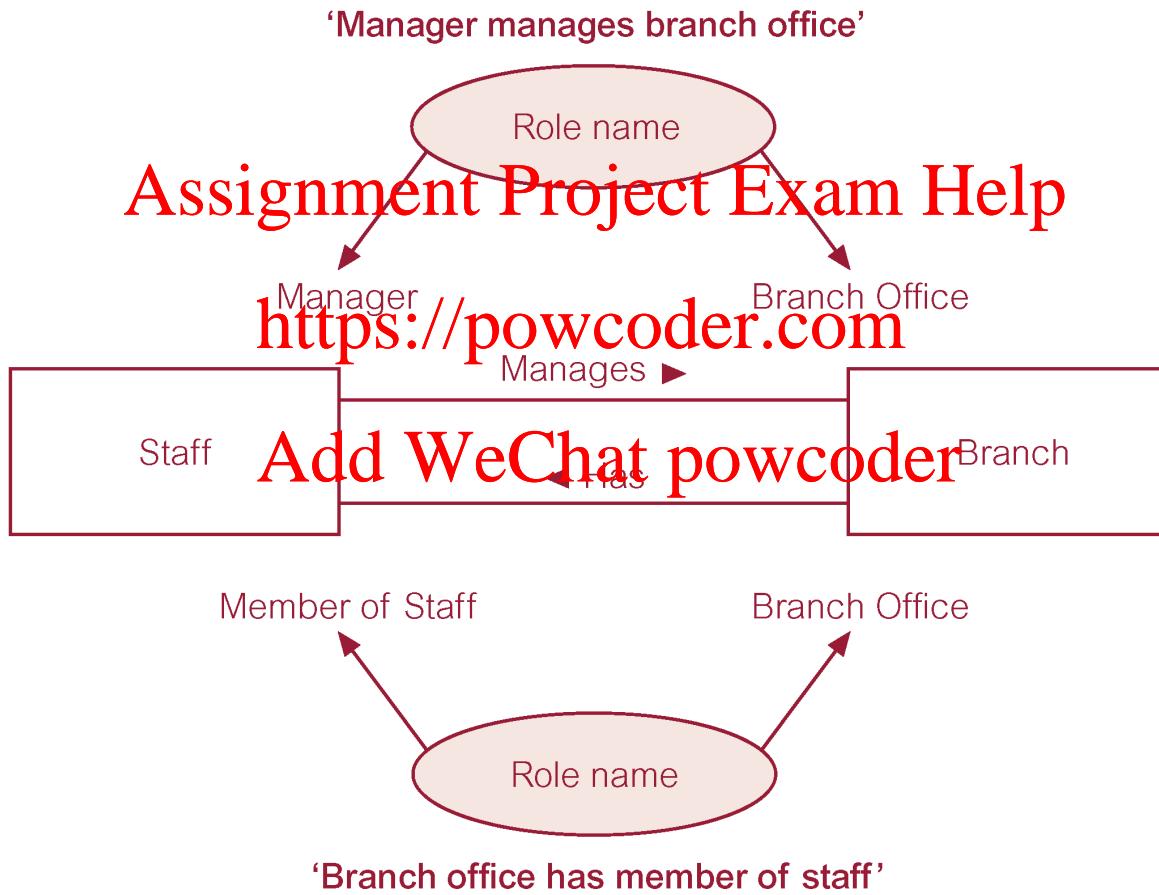
- Recursive Relationship
  - Relationship type where **same** entity type participates more than **Assignment Project Exam Help**
- Relationships may be given role names to indicate purpose that each participating entity type plays in a relationship. **Add WeChat powcoder**

# Recursive Relationship Called Supervises with Role Names

'Staff (Supervisor) supervises staff (Supervisee)' **Assignment Project Exam Help**



# Entities Associated Through Two Distinct Relationships with Role Names



# Attributes (1 of 4)

- Attribute
  - Property of an entity or a relationship type.
- Attribute Domain
  - Set of allowable values for one or more attributes.

Add WeChat powcoder

## Attributes (2 of 4)

- Simple Attribute
  - Attribute composed of a single component with an independent existence.
- Composite Attribute
  - Attribute composed of multiple components, each with an independent existence.

## Attributes (3 of 4)

- Single-valued Attribute
  - Attribute that holds a single value for each occurrence of an entity type.
- Multi-valued Attribute
  - Attribute that holds multiple values for each occurrence of an entity type.

# Attributes (4 of 4)

- Derived Attribute
  - Attribute that represents a value that is derivable from value of ~~Assignment~~, ~~Project~~, ~~Exam~~, ~~Help~~ attributes, not necessarily in the same entity type.

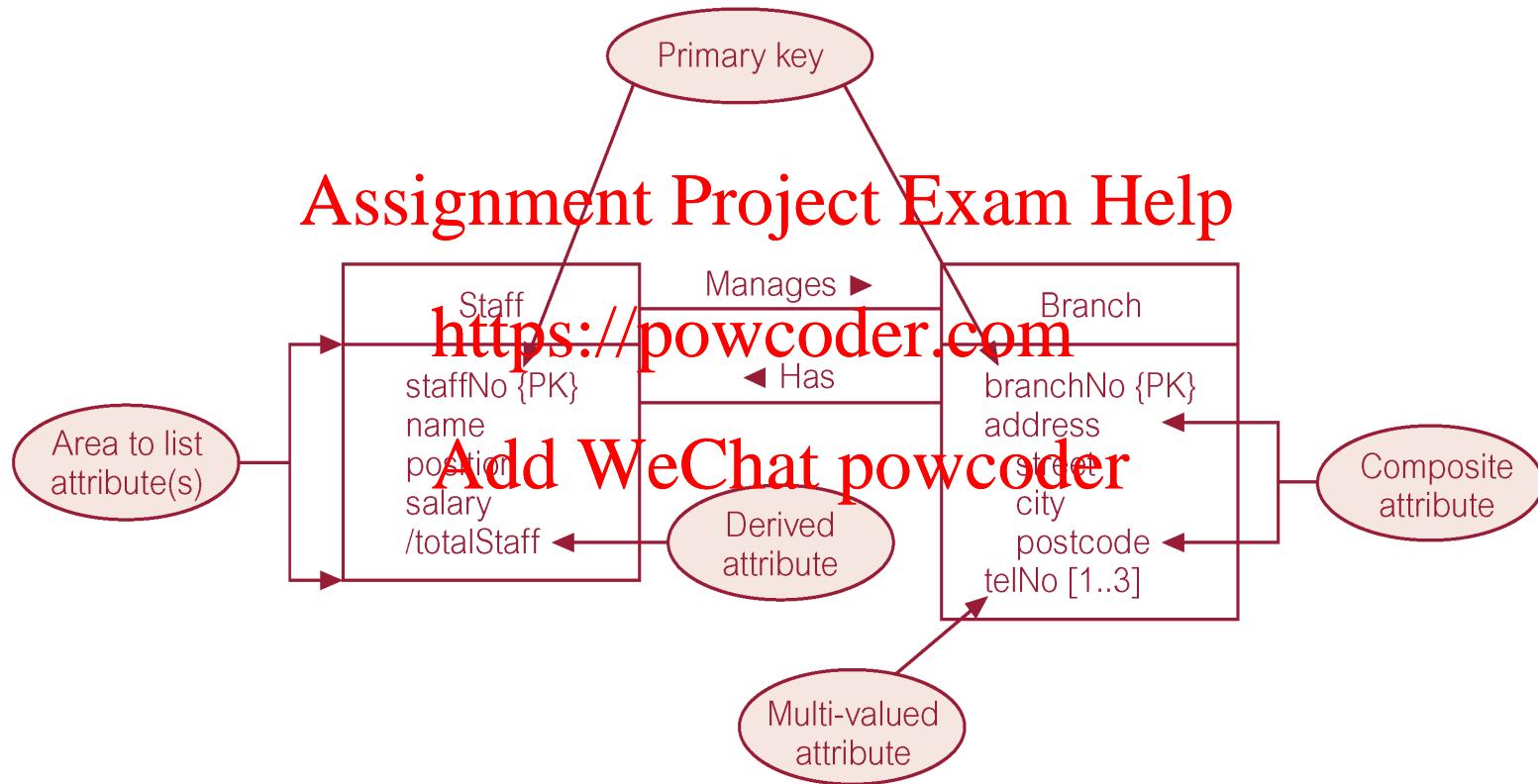
<https://powcoder.com>

Add WeChat powcoder

# Keys

- Candidate Key
  - Minimal set of attributes that uniquely identifies each occurrence of an entity type.
- Primary Key
  - Candidate key selected to uniquely identify each occurrence of an entity type.
- Composite Key
  - A candidate key that consists of two or more attributes.

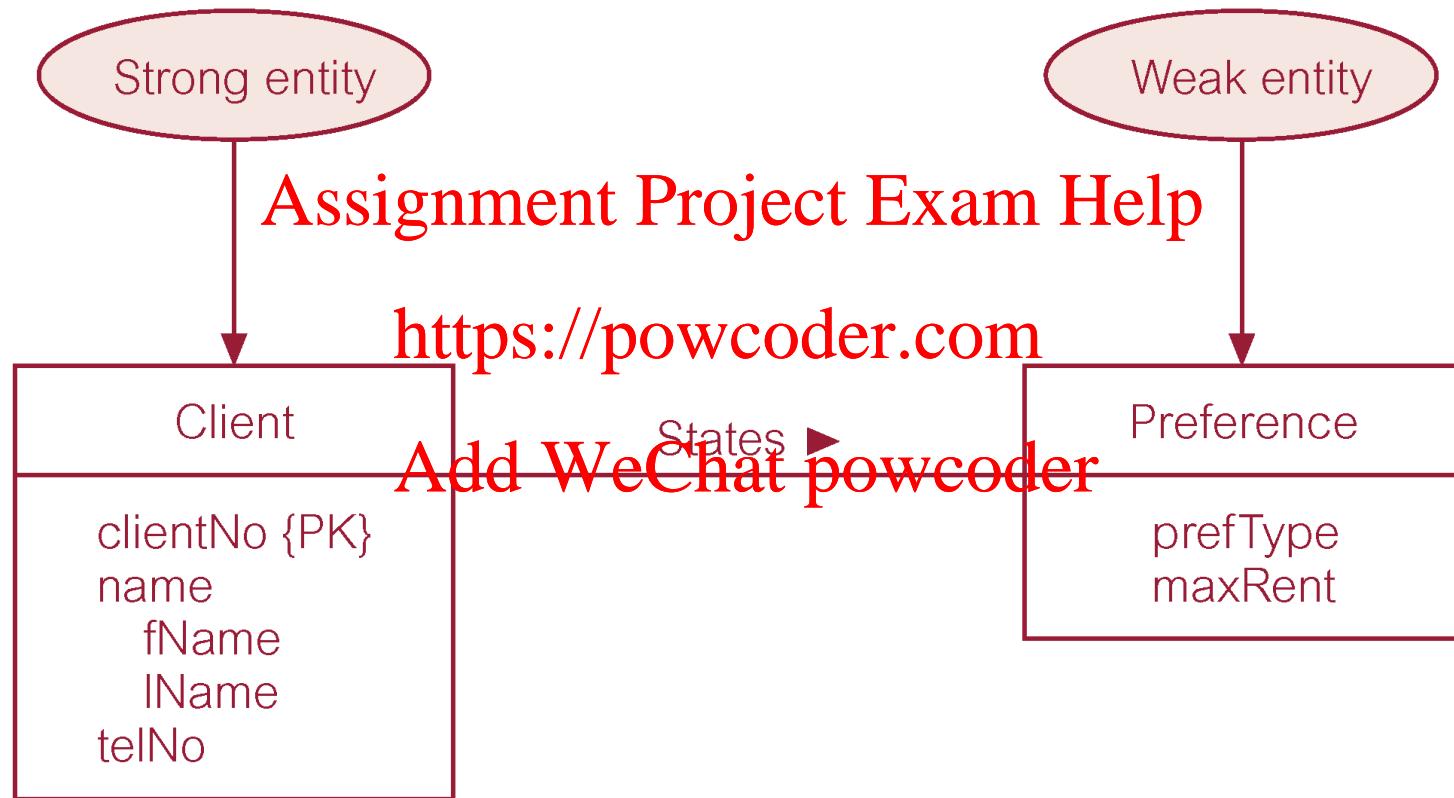
# ER Diagram of Staff and Branch Entities and Their Attributes



# Entity Type (2 of 2)

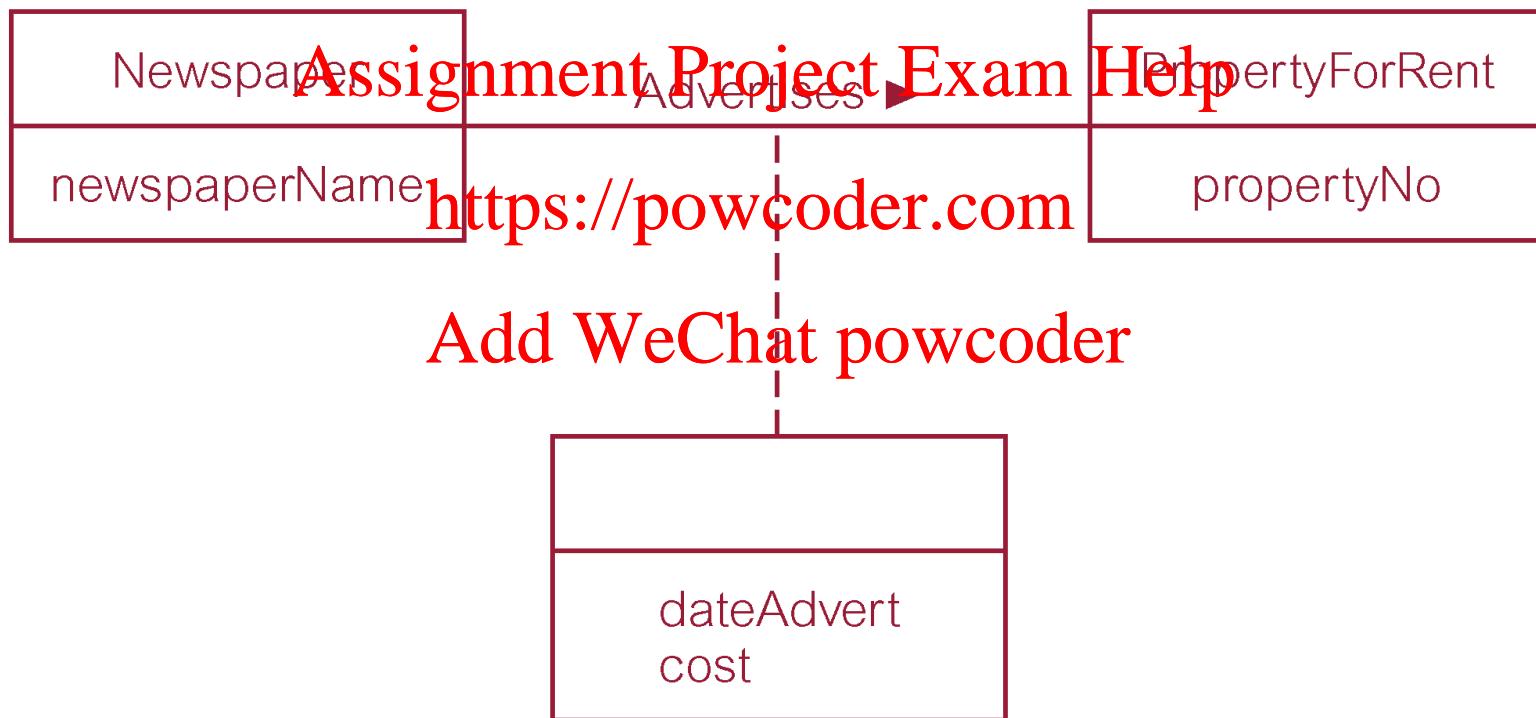
- Strong Entity Type
  - Entity type that is **not** existence-dependent on some other entity type.
- Weak Entity Type
  - Entity type that is existence-dependent on some other entity type.

# Strong Entity Type Called Client and Weak Entity Type Called Preference



# Relationship Called Advertises with Attributes

## ‘Newspaper advertises property for rent’



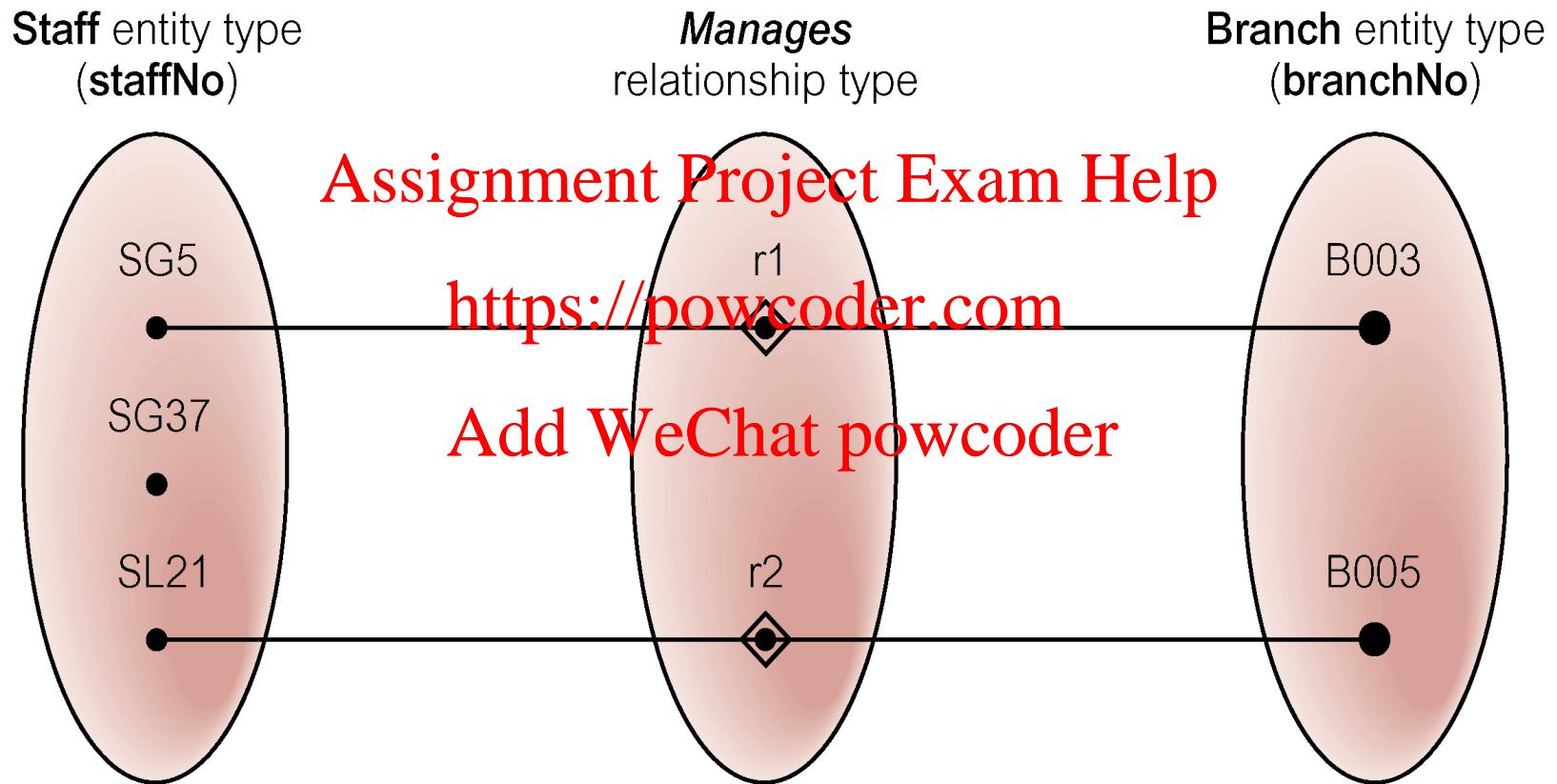
# Structural Constraints (1 of 5)

- Main type of constraint on relationships is called **multiplicity**.
- Multiplicity - number (or range) of possible occurrences of an entity type that may relate to a single occurrence of an associated entity type through a particular relationship.
- Represents policies (called **business rules**) established by user or company.

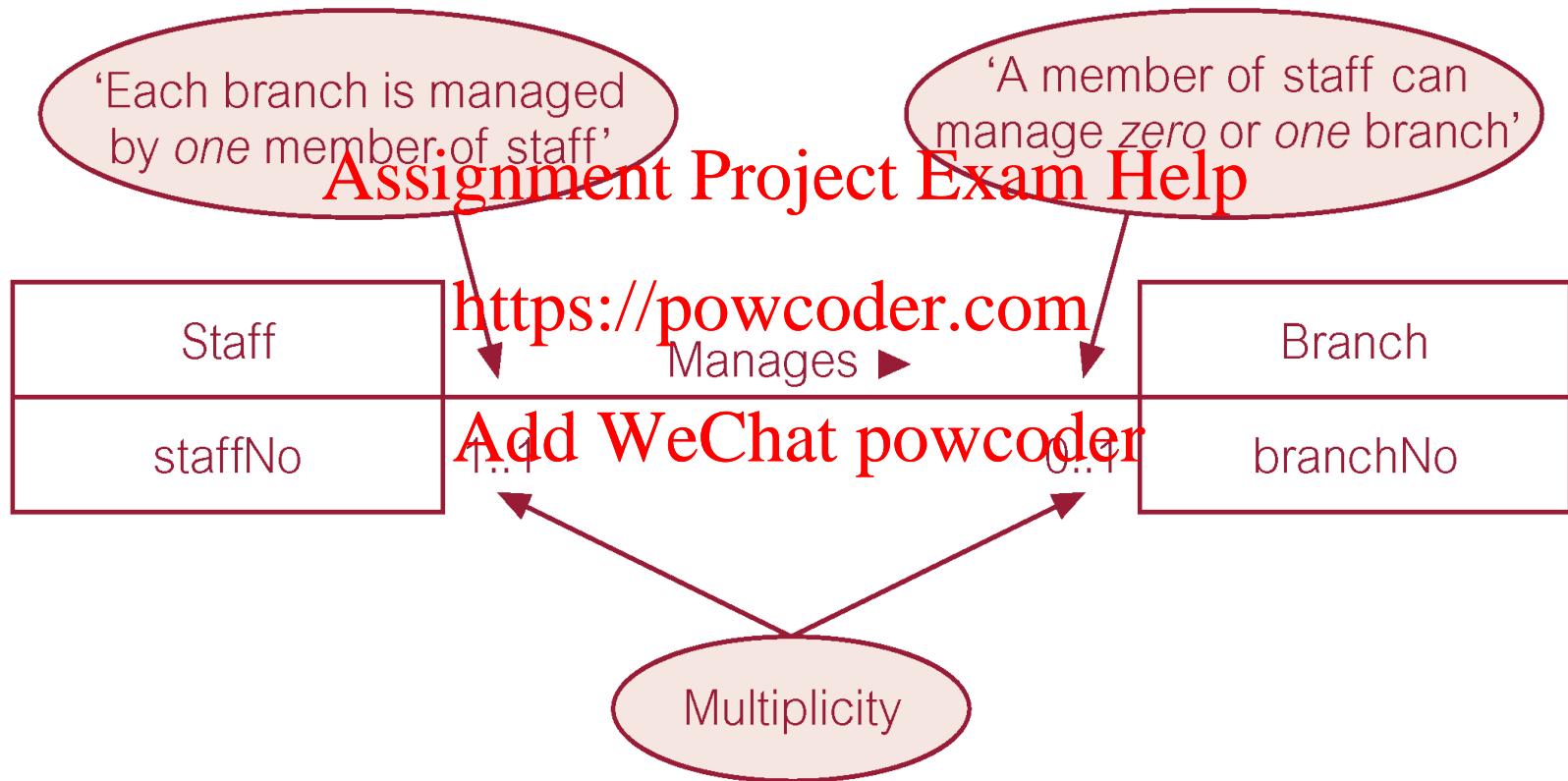
# Structural Constraints (2 of 5)

- The most common degree for relationships is binary.
- Binary relationships are generally referred to as being:
  - one-to-one (1:1)
  - one-to-many ( $1:\ast$ )
  - many-to-many ( $\ast:\ast$ )

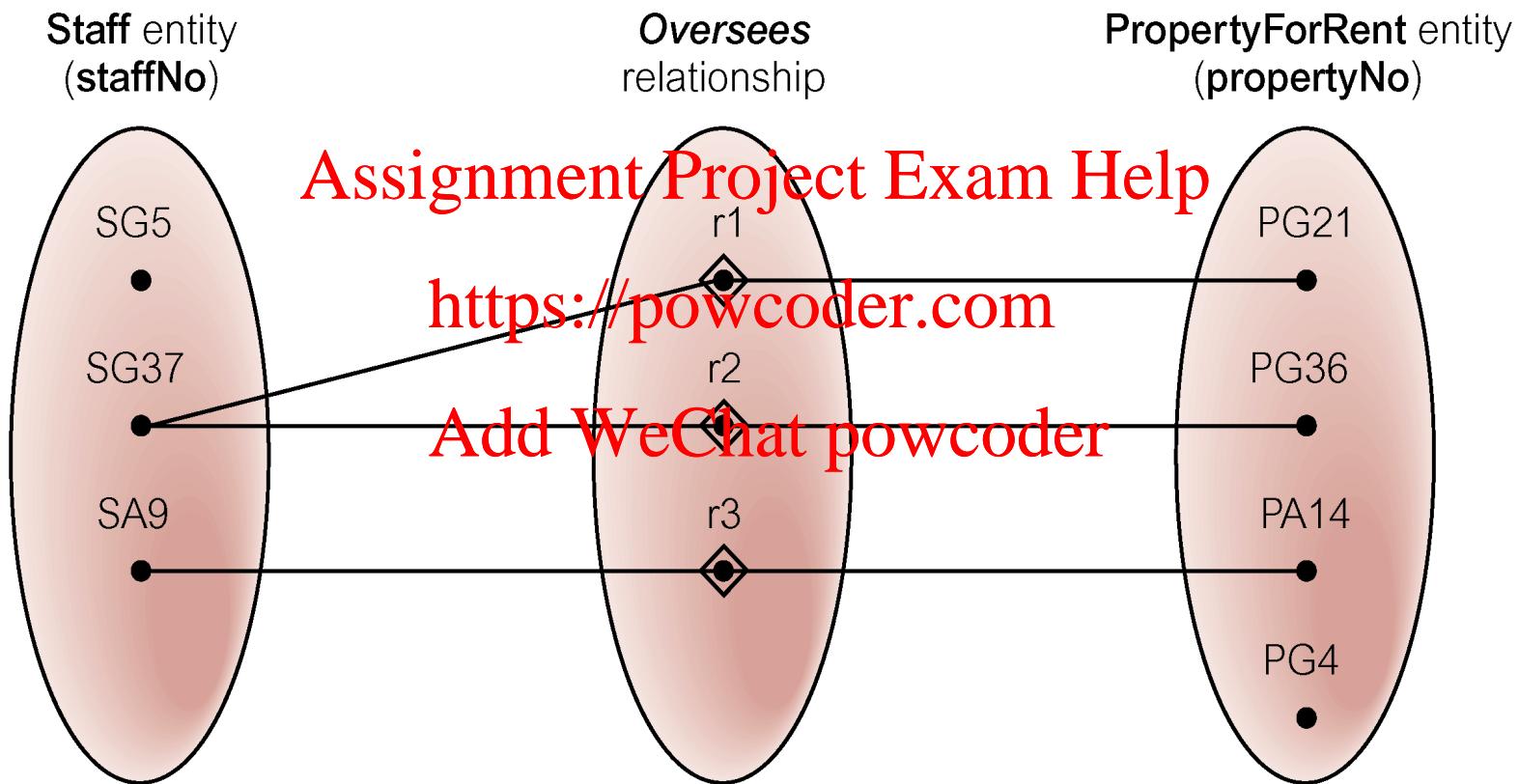
# Semantic Net of Staff Manages Branch Relationship Type



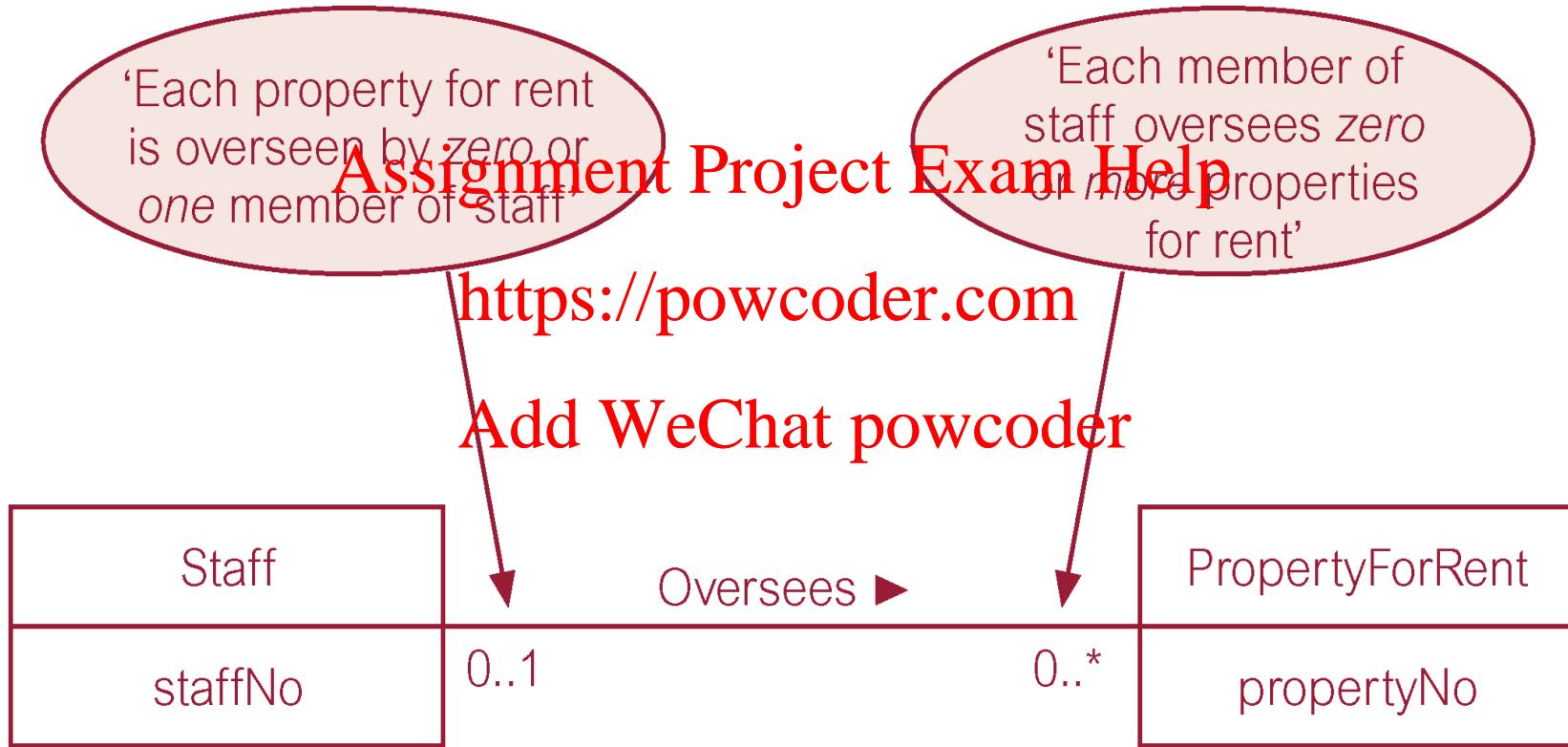
# Multiplicity of Staff Manages Branch (1 : 1) Relationship



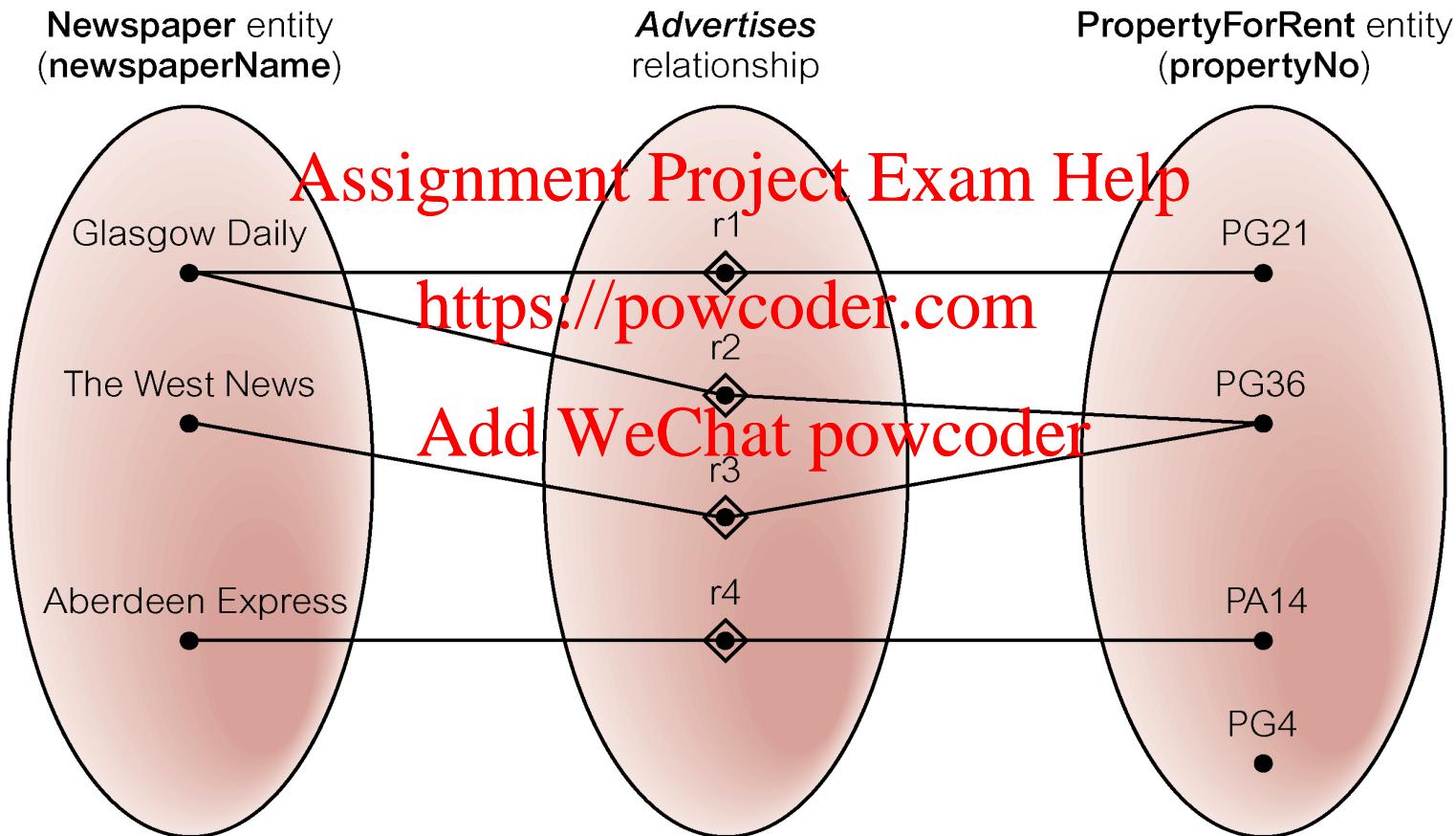
# Semantic Net of Staff Oversees PropertyForRent Relationship Type



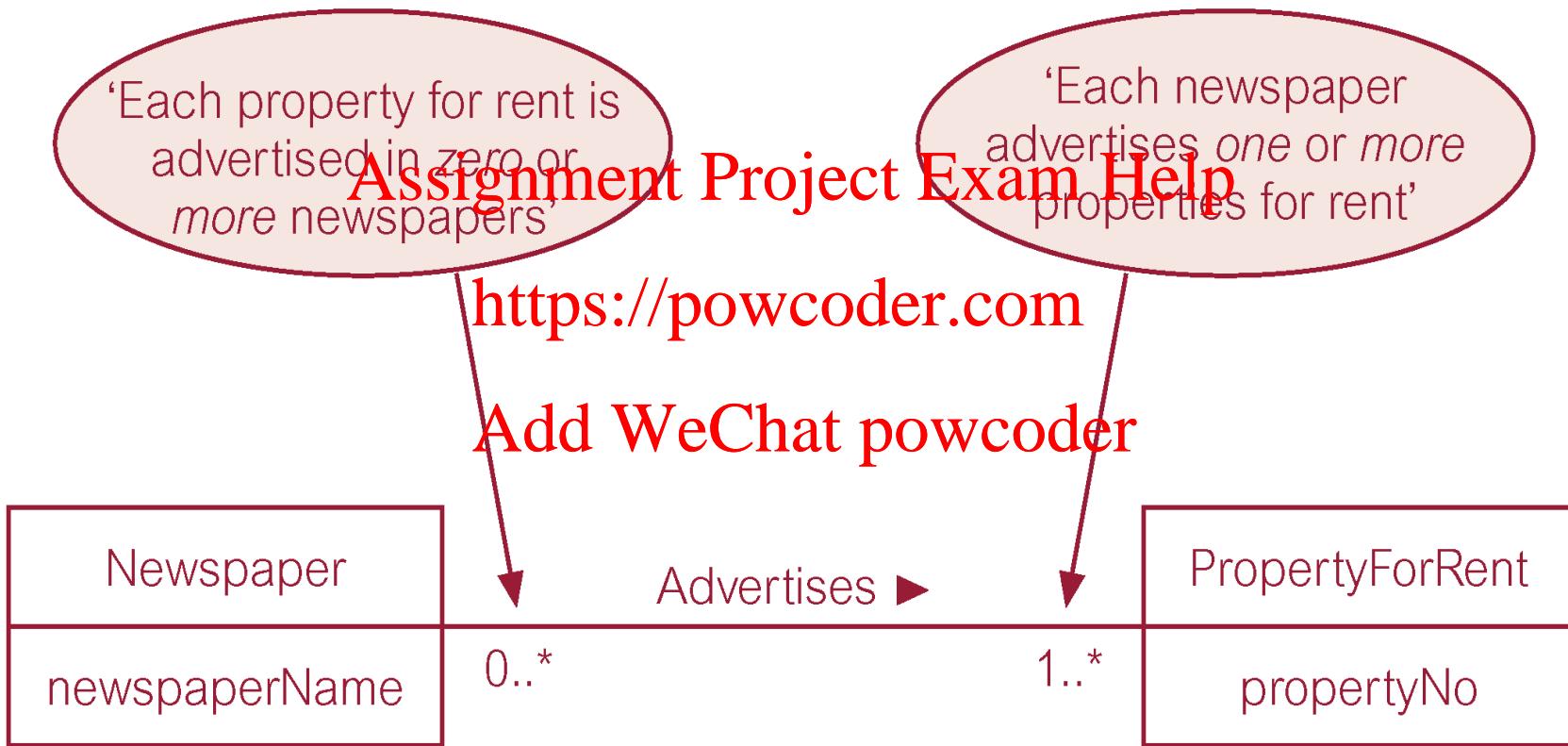
# Multiplicity of Staff Oversees PropertyForRent (1 : \*) Relationship Type



# Semantic Net of Newspaper Advertises PropertyForRent Relationship Type



# Multiplicity of Newspaper Advertises PropertyForRent (\* : \*) Relationship



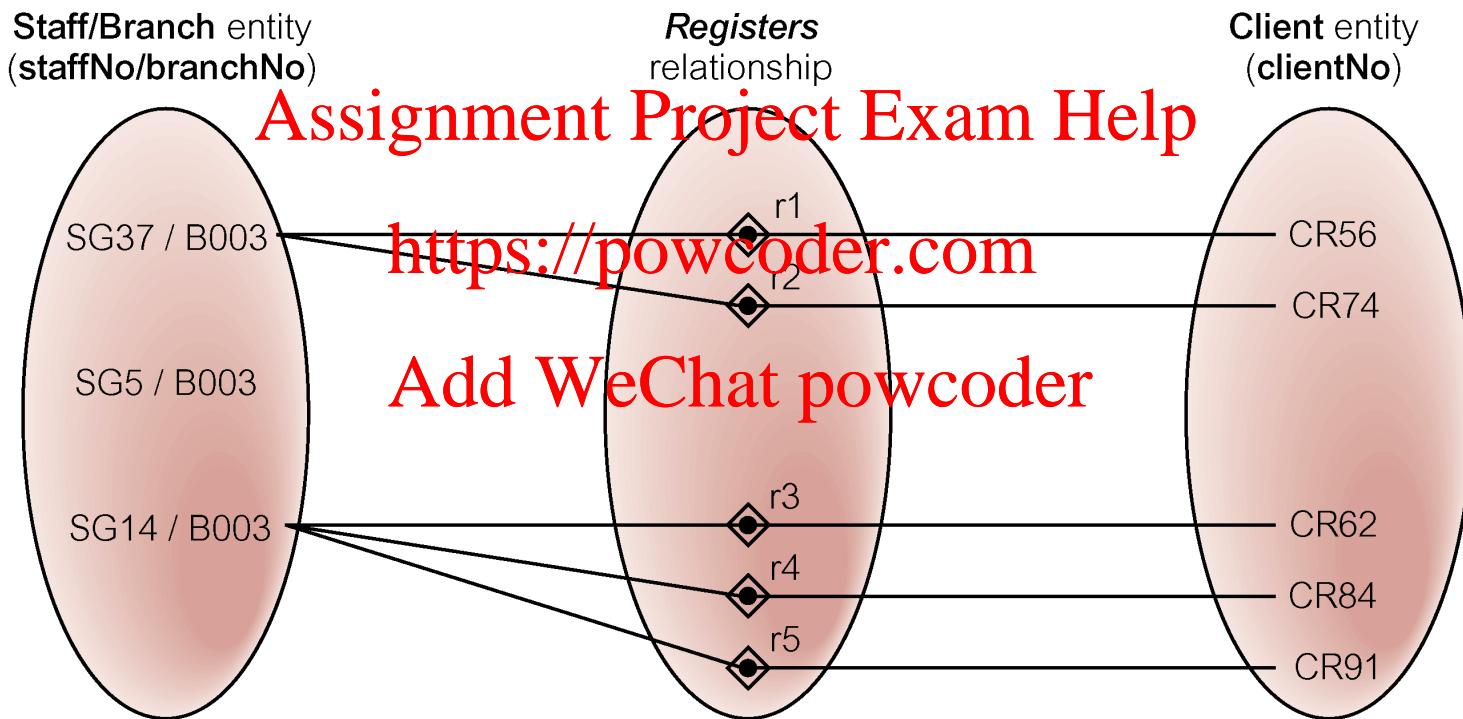
# Structural Constraints (3 of 5)

- Multiplicity for Complex Relationships
  - Number (or range) of possible occurrences of an entity type in a many-to-many relationship (n-1) values are fixed.

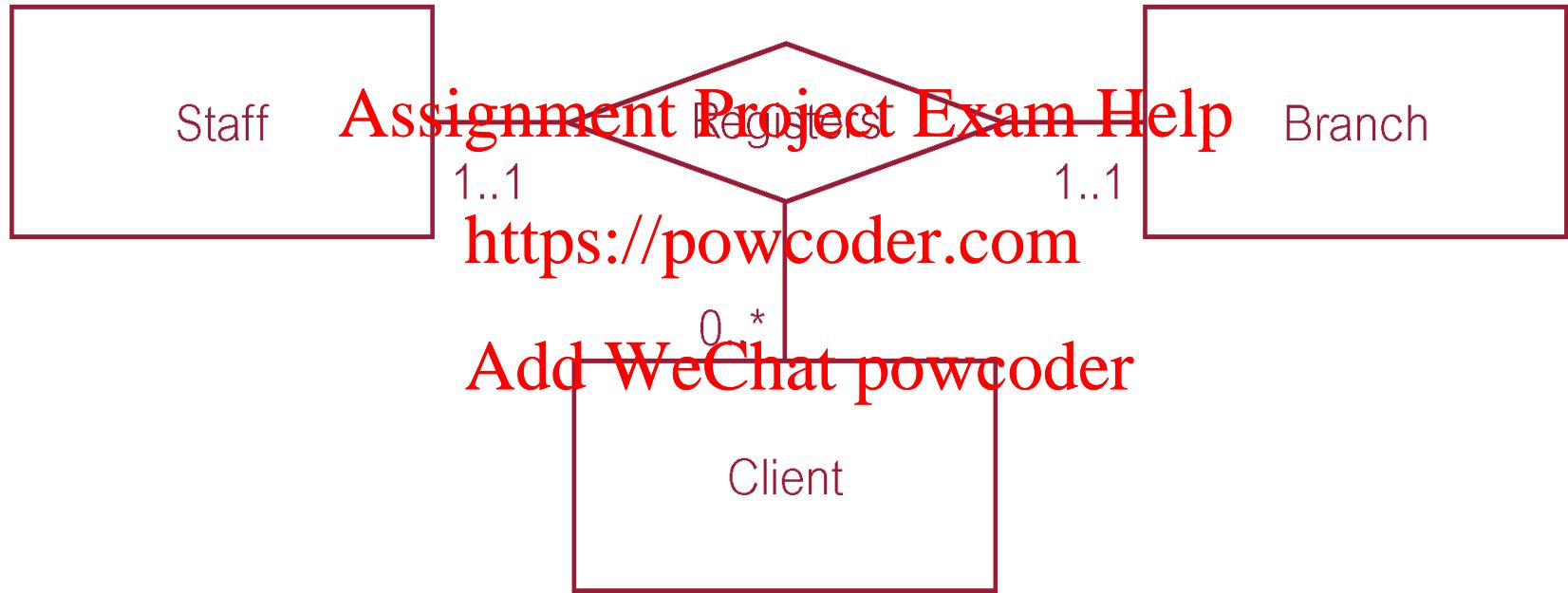
<https://powcoder.com>

Add WeChat powcoder

# Semantic Net of Ternary Registers Relationship with Values for Staff and Branch Entities Fixed



# Multiplicity of Ternary Registers Relationship



# Summary of Multiplicity Constraints

Alternative Ways To Represent Multiplicity Constraints	Meaning
0..1	Zero or one entity occurrence
1..1(or just 1)	Exactly one entity occurrence
0.. * (or just *)	Zero or many entity occurrences
1.. *	One or many entity occurrences
5..10	Minimum of 5 up to a maximum of 10 entity occurrences
0, 3, 6–8	Zero or three or six, seven, or eight entity occurrences

# Structural Constraints (4 of 5)

- Multiplicity is made up of two types of restrictions on relationships: **cardinality** and **participation**.

Assignment Project Exam Help

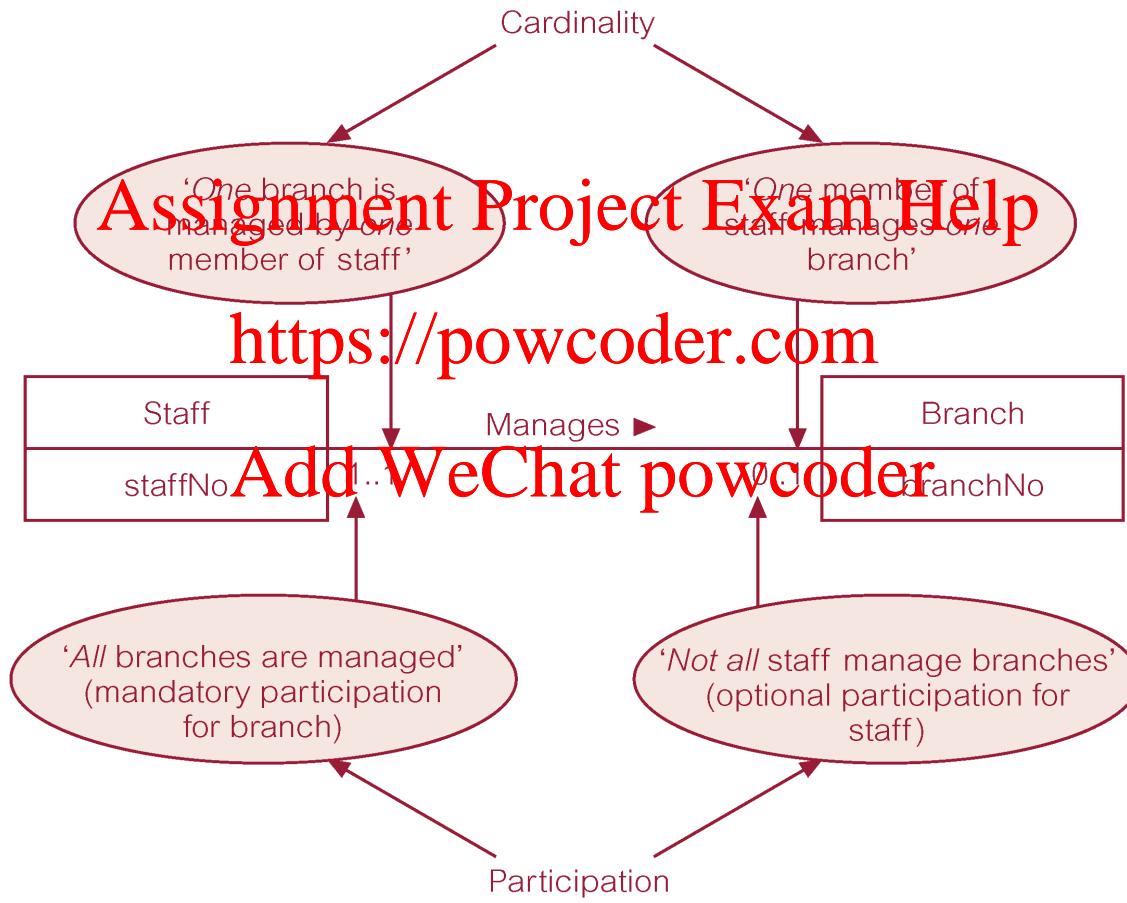
<https://powcoder.com>

Add WeChat powcoder

# Structural Constraints (5 of 5)

- Cardinality
  - Describes maximum number of possible relationship occurrences for an entity participating in a given relationship type.  
<https://powcoder.com>
- Participation
  - Determines whether all or only some entity occurrences participate in a relationship.

# Multiplicity as Cardinality and Participation Constraints



# Problems with ER Models (1 of 2)

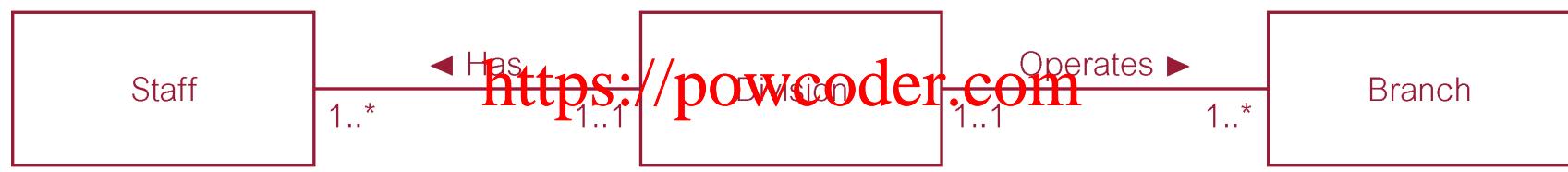
- Problems may arise when designing a conceptual data model called **connection traps**.
- Often due to a misinterpretation of the meaning of certain relationships. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com>
- Two main types of connection traps are called **fan traps** and **chasm traps**. [Add WeChat powcoder](#)

# Problems with ER Models (2 of 2)

- Fan Trap
  - Where a model represents a relationship between entity types, but pathway between certain entity occurrences is ambiguous.  
<https://powcoder.com>
- Chasm Trap
  - Where a model suggests the existence of a relationship between entity types, but pathway does not exist between certain entity occurrences.

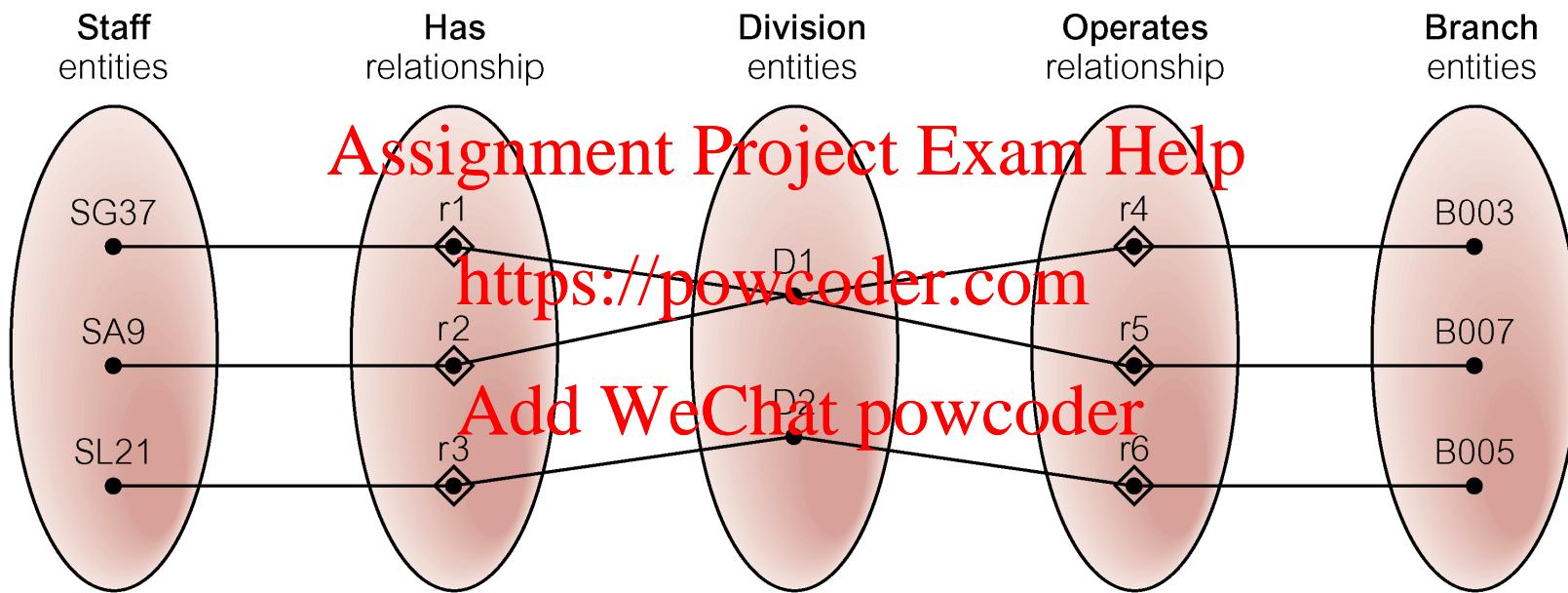
# An Example of a Fan Trap

Assignment Project Exam Help



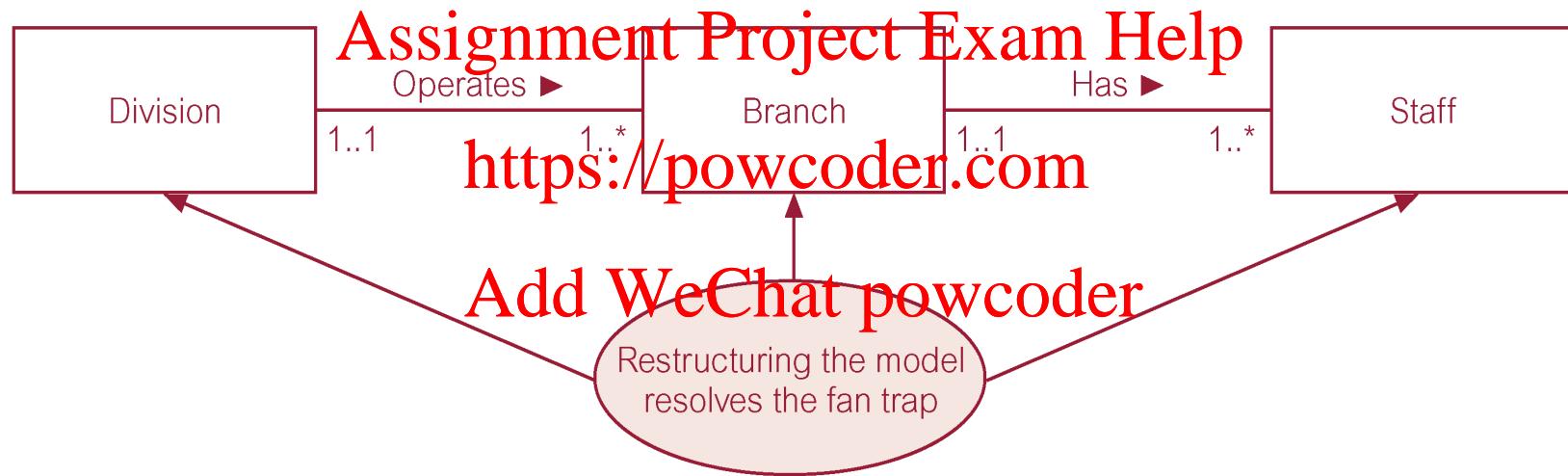
Add WeChat powcoder

# Semantic Net of ER Model with Fan Trap

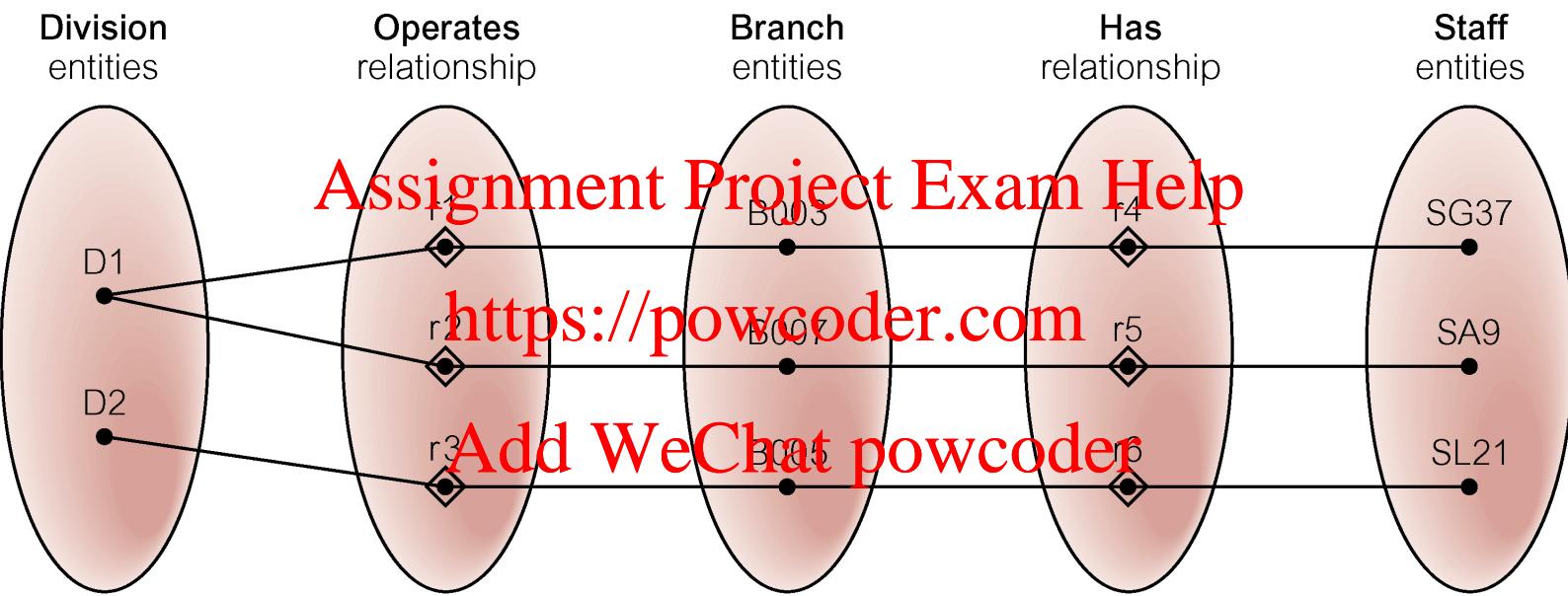


- At which branch office does staff number SG37 work?

# Restructuring ER Model to Remove Fan Trap

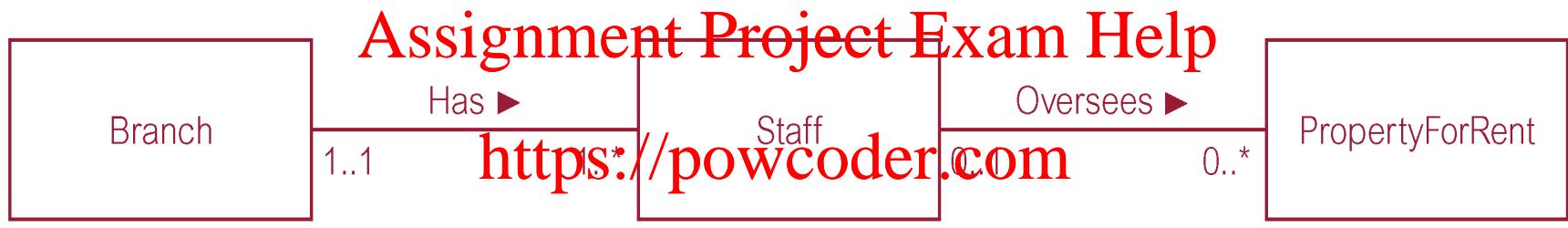


# Semantic Net of Restructured ER Model with Fan Trap Removed



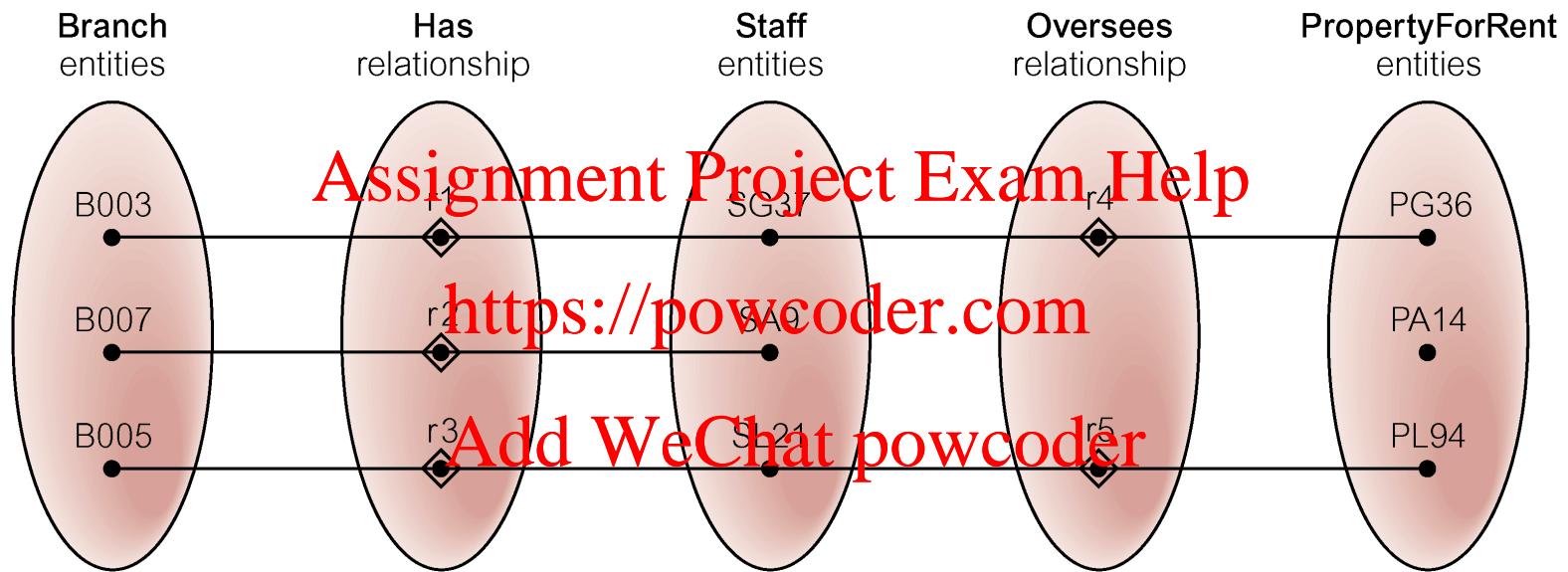
- SG37 works at branch B003.

# An Example of a Chasm Trap



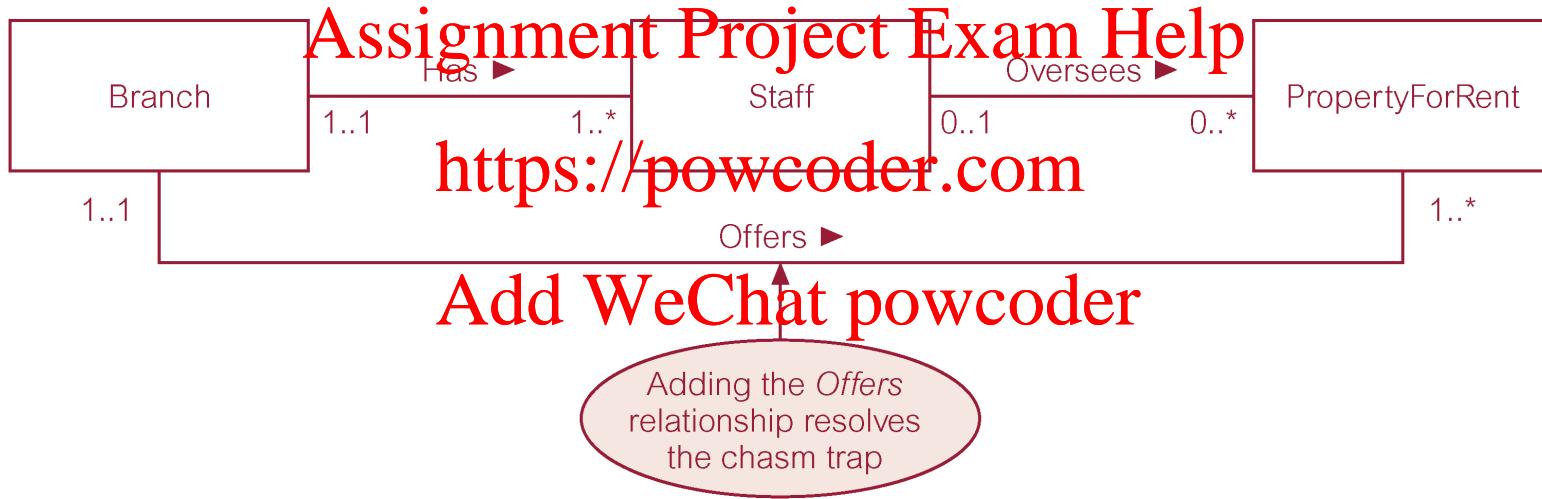
Add WeChat powcoder

# Semantic Net of ER Model with Chasm Trap

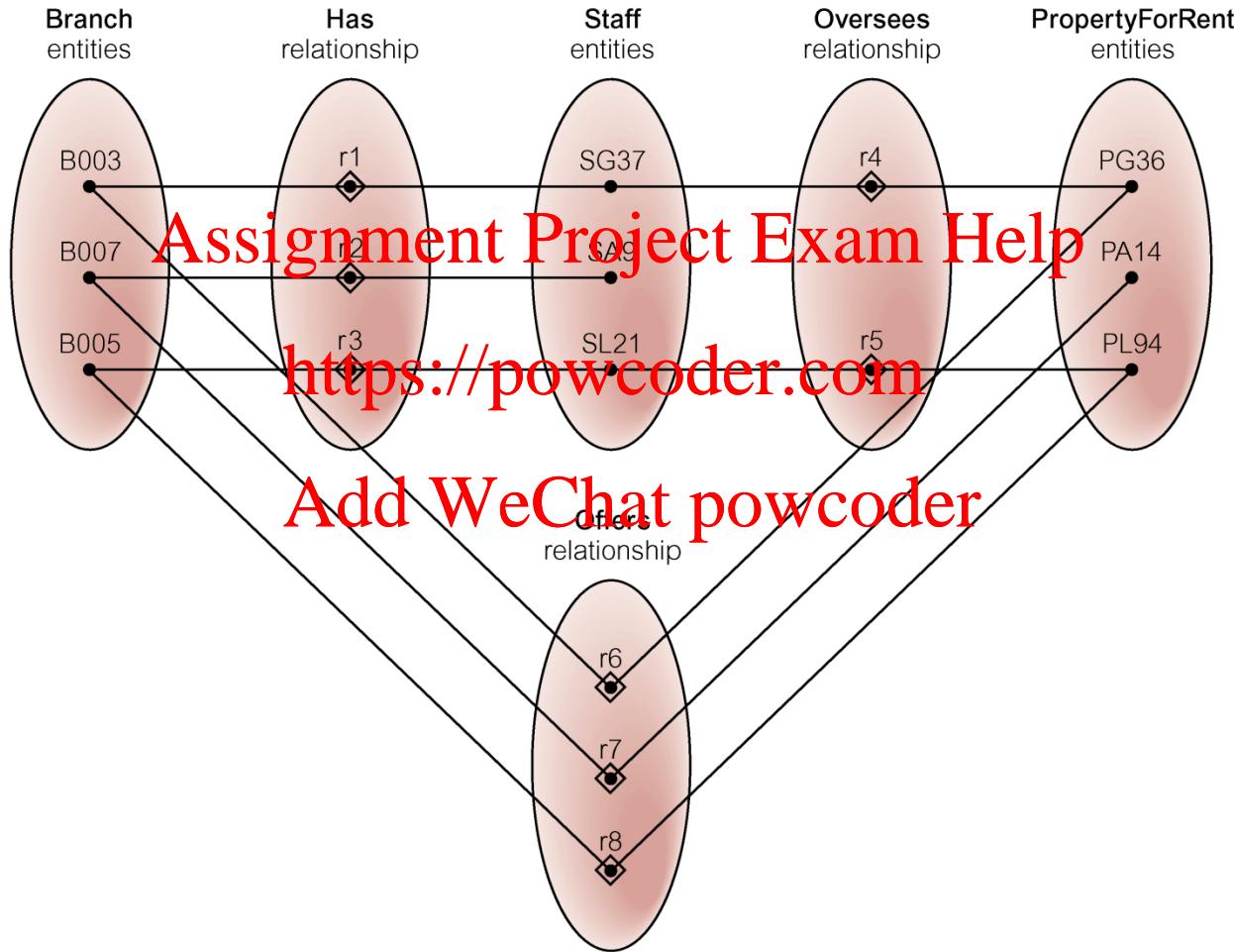


- At which branch office is property PA14 available?

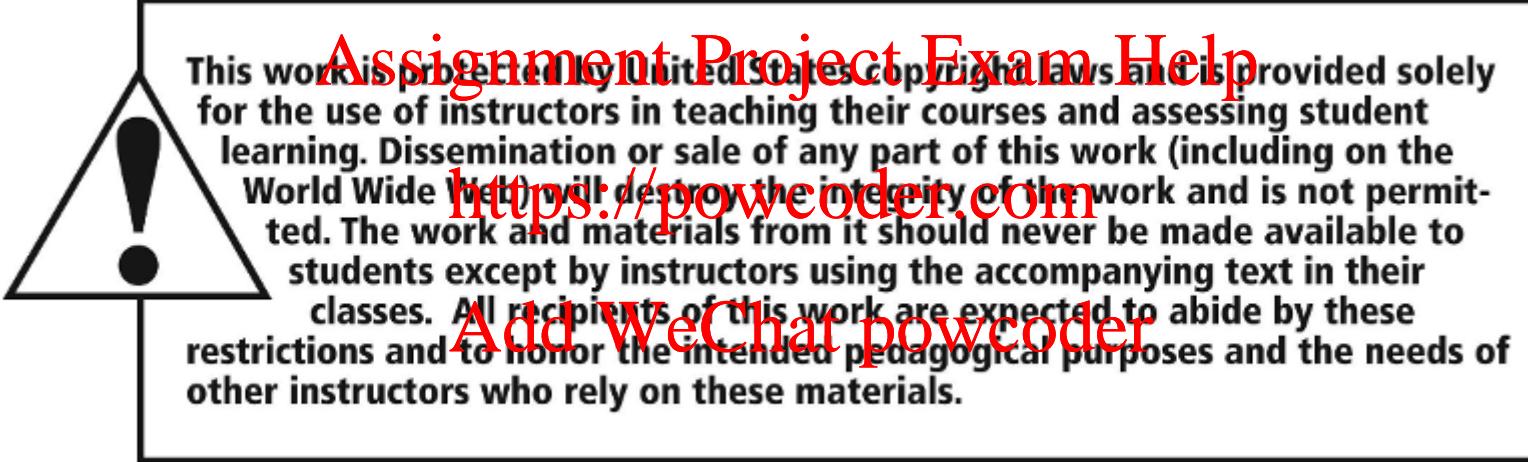
# ER Model Restructured to Remove Chasm Trap



# Semantic Net of Restructured ER Model with Chasm Trap Removed



# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 13

<https://powcoder.com>  
Enhanced Entity-Relationship  
Modeling

Add WeChat powcoder

# Learning Objectives

**13.1** Limitations of basic concepts of the ER model and requirements to represent more complex applications using additional data modeling concepts  
**Assignment Project Exam Help**

**13.2** Most useful additional data modeling concept of Enhanced ER (EER) model is called specialization/generalization  
**Add WeChat powcoder**

**13.3** A diagrammatic technique for displaying specialization/generalization in an EER diagram using UML.

# Enhanced Entity-Relationship Model

- Since 1980s there has been an increase in emergence of new database applications with more demanding requirements  
**Assignment Project Exam Help**
- Basic concepts of ER modeling are not sufficient to represent requirements of newer, more complex applications.  
**Add WeChat powcoder**
- Response is development of additional ‘semantic’ modeling concepts.

# The Enhanced Entity-Relationship Model

- Semantic concepts are incorporated into the original ER model and called the Enhanced Entity-Relationship (EER) model.
- Examples of additional concept of EER model is called specialization / generalization.

[Assignment Project Exam Help](https://powcoder.com)

<https://powcoder.com>

[Add WeChat powcoder](https://powcoder.com)

# Specialization / Generalization (1 of 4)

- Superclass
  - An entity type that includes one or more distinct subgroups
- Subclass
  - A distinct subgrouping of occurrences of an entity type.

[Assignment](#) [Project](#) [Exam](#) [Help](#)

<https://powcoder.com>

[Add WeChat powcoder](#)

# Specialization / Generalization (2 of 4)

- Superclass/subclass relationship is one-to-one (1:1).
- Superclass may contain overlapping or distinct subclasses.
- Not all members of a superclass need be a member of a subclass.

Add WeChat powcoder

# Specialization / Generalization (3 of 4)

- Attribute Inheritance
  - An entity in a subclass represents same ‘real world’ object as ~~Assignment Project Exam Help~~ and may possess subclass-specific attributes, as well as those associated with the superclass.

Add WeChat powcoder

# Specialization / Generalization (4 of 4)

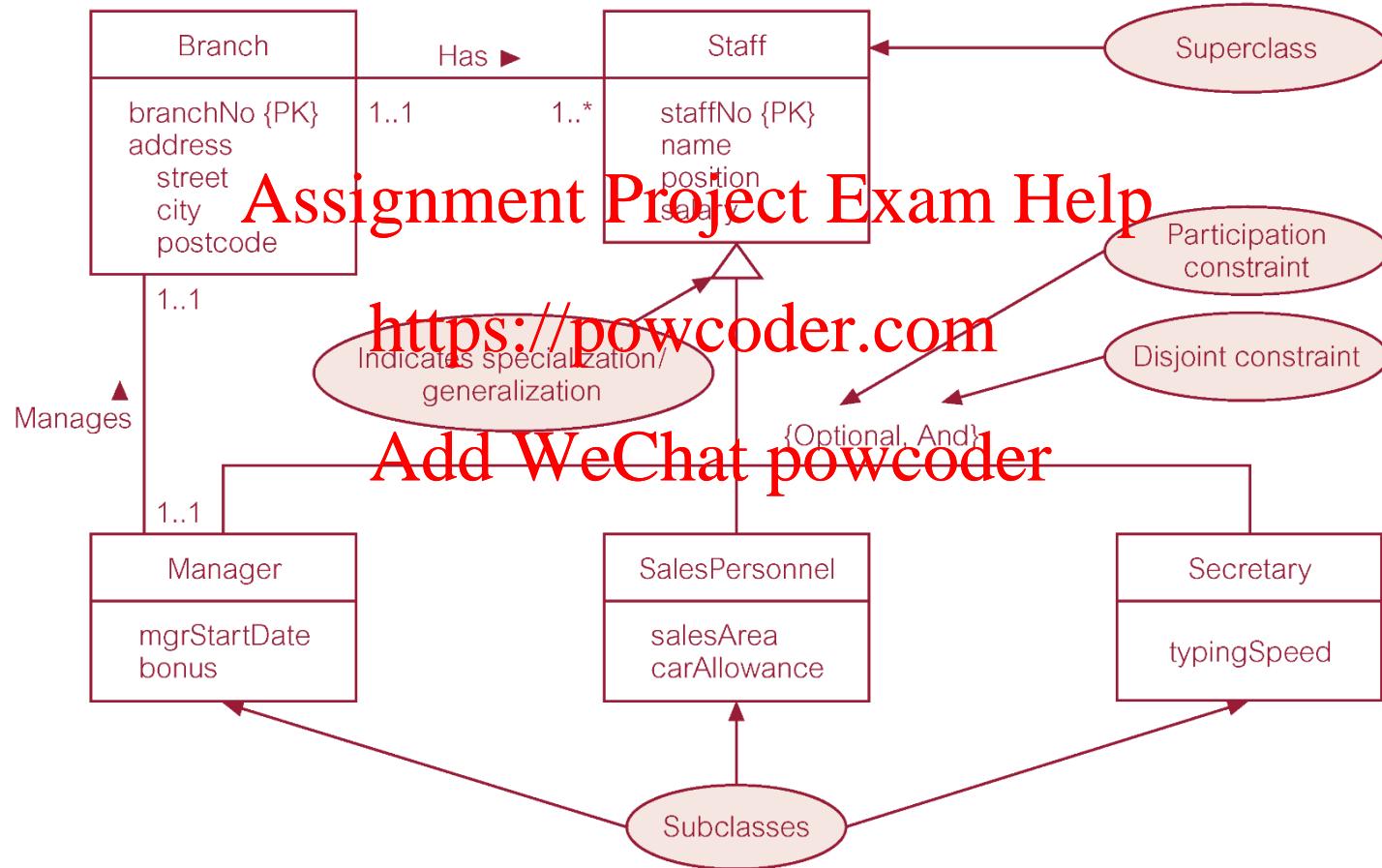
- Specialization
  - Process of maximizing differences between members of an entity by identifying their distinguishing characteristics.
- Generalization
  - Process of minimizing differences between entities by identifying their common characteristics.

<https://powcoder.com>

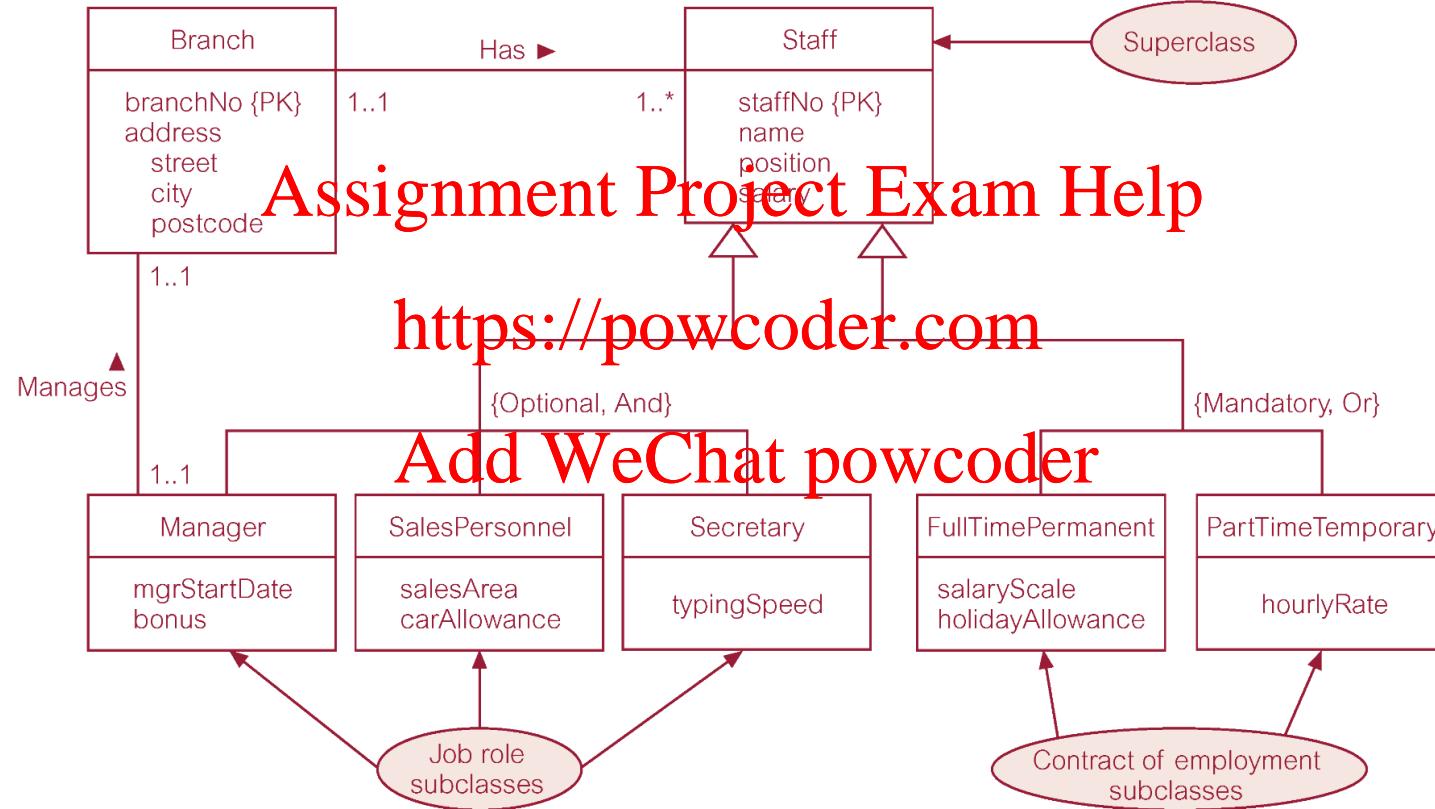
# AllStaff Relation Holding Details of All Staff

staffNo	name	position	salary	mgrStartDate	bonus	sales Area	car Allowance	typing Speed
SL21	John White	Manager	30000	1/02/95	1000			
SG37	Ann Beech	Assistant	12000					
SG66	Mary Martinez	Sales Manager	27000			SA1A	5000	
SA9	Mary Howe	Assistant	9000					
SL89	Stuart Stern	Secretary	8500					100
SL31	Robert Chin	Snr Sales Asst	17000			SA2B	3700	
SG5	Susan Brand	Manager	24000	01/06/91	2350			

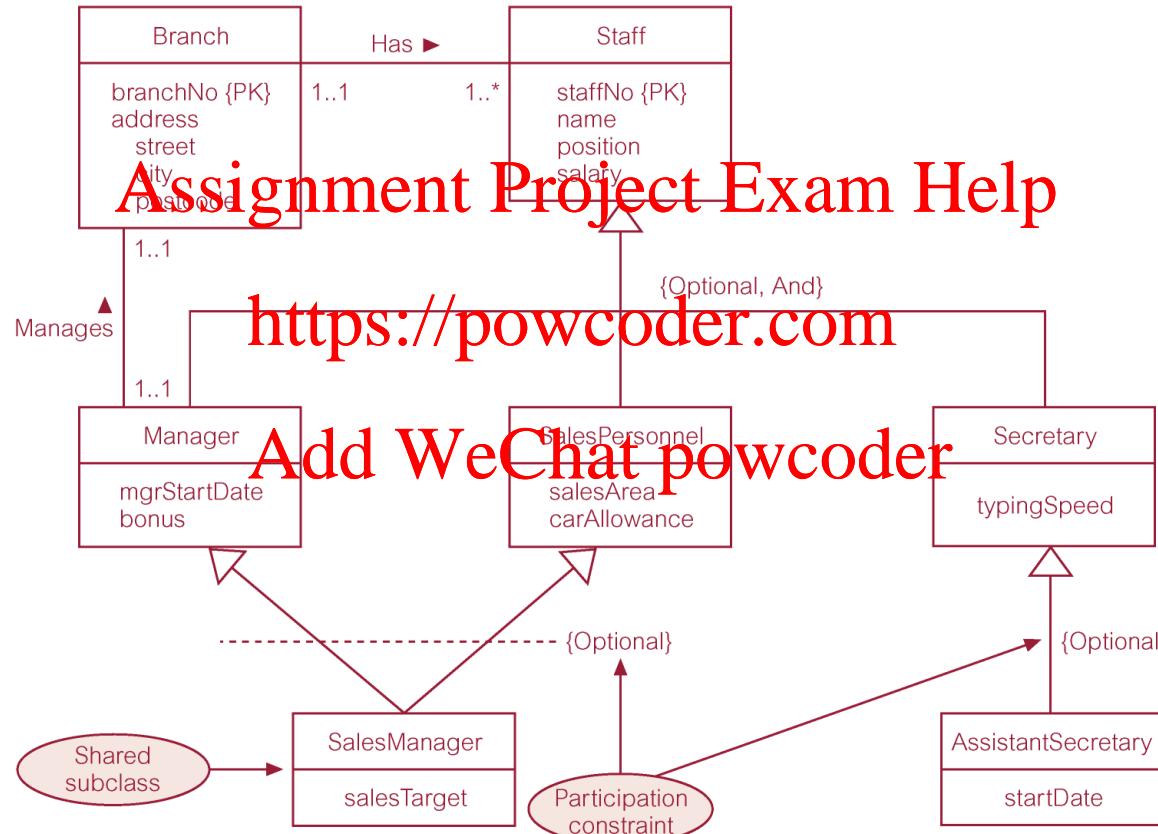
# Specialization/Generalization of Staff Entity into Subclasses Representing Job Roles



# Specialization/Generalization of Staff Entity into Job Roles and Contracts of Employment



# EER Diagram with Shared Subclass and Subclass with Its Own Subclass



# Constraints on Specialization/Generalization (1 of 3)

- Two constraints that may apply to a specialization/generalization:
  - participation constraints
  - disjoint constraints
- Participation constraint
  - Determines whether every member in superclass must participate as a member of a subclass.
  - May be **mandatory or optional**.

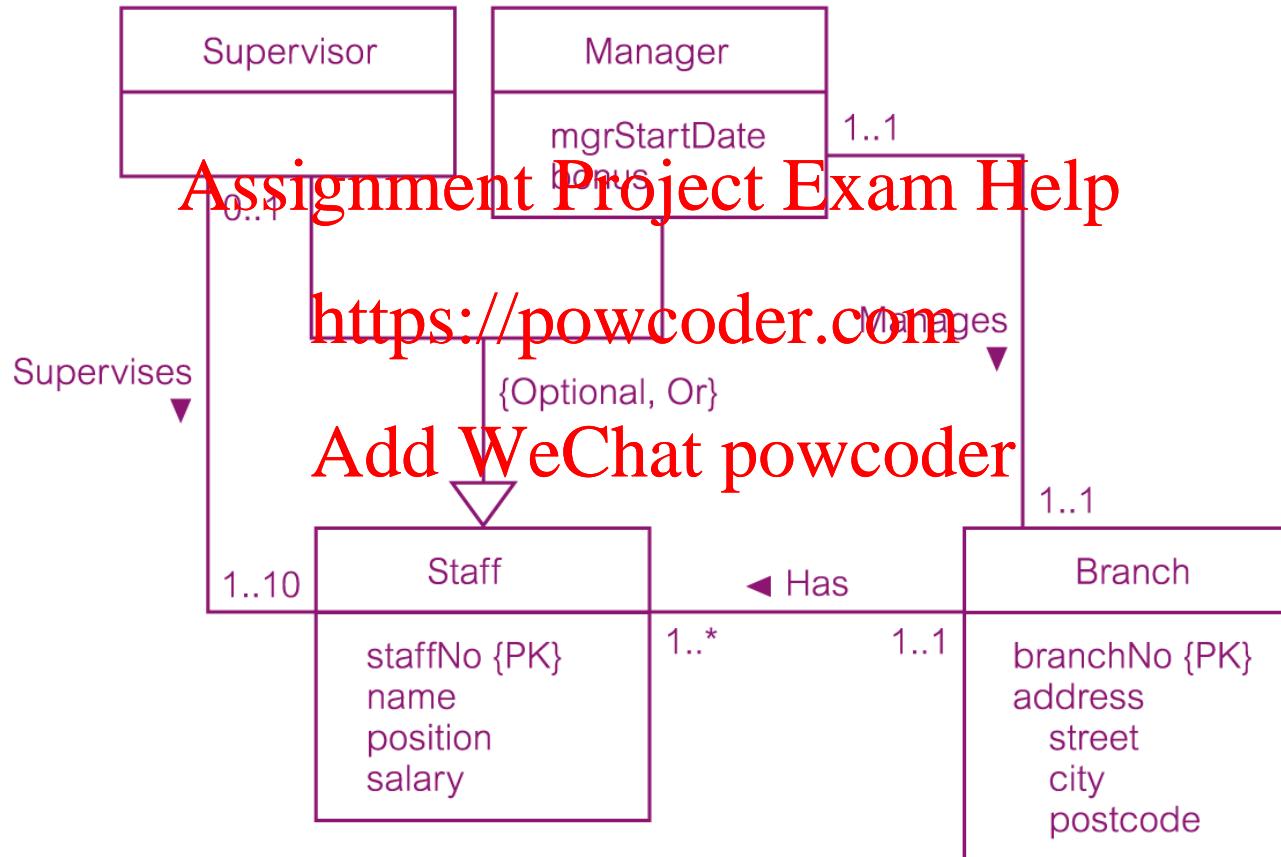
## Constraints on Specialization/Generalization (2 of 3)

- Disjoint constraint
  - Describes relationship between members of the subclasses and indicates whether a member of a superclass can be a member of one, or more than one, subclass.
  - May be disjoint or nondisjoint

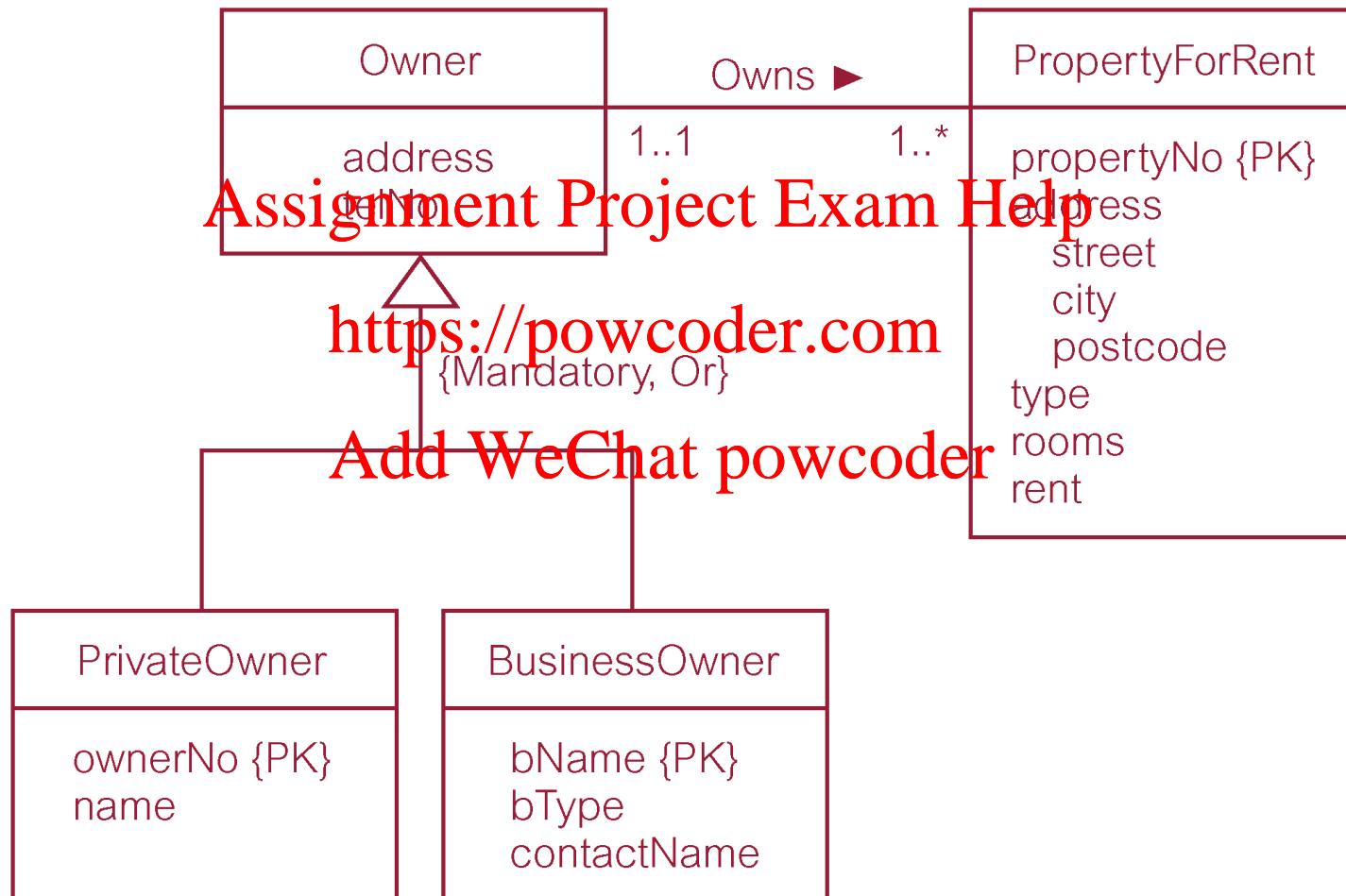
## Constraints on Specialization/Generalization (3 of 3)

- There are four categories of constraints of specialization and generalization:
  - mandatory and disjoint
  - optional and disjoint
  - mandatory and nondisjoint
  - optional and nondisjoint.

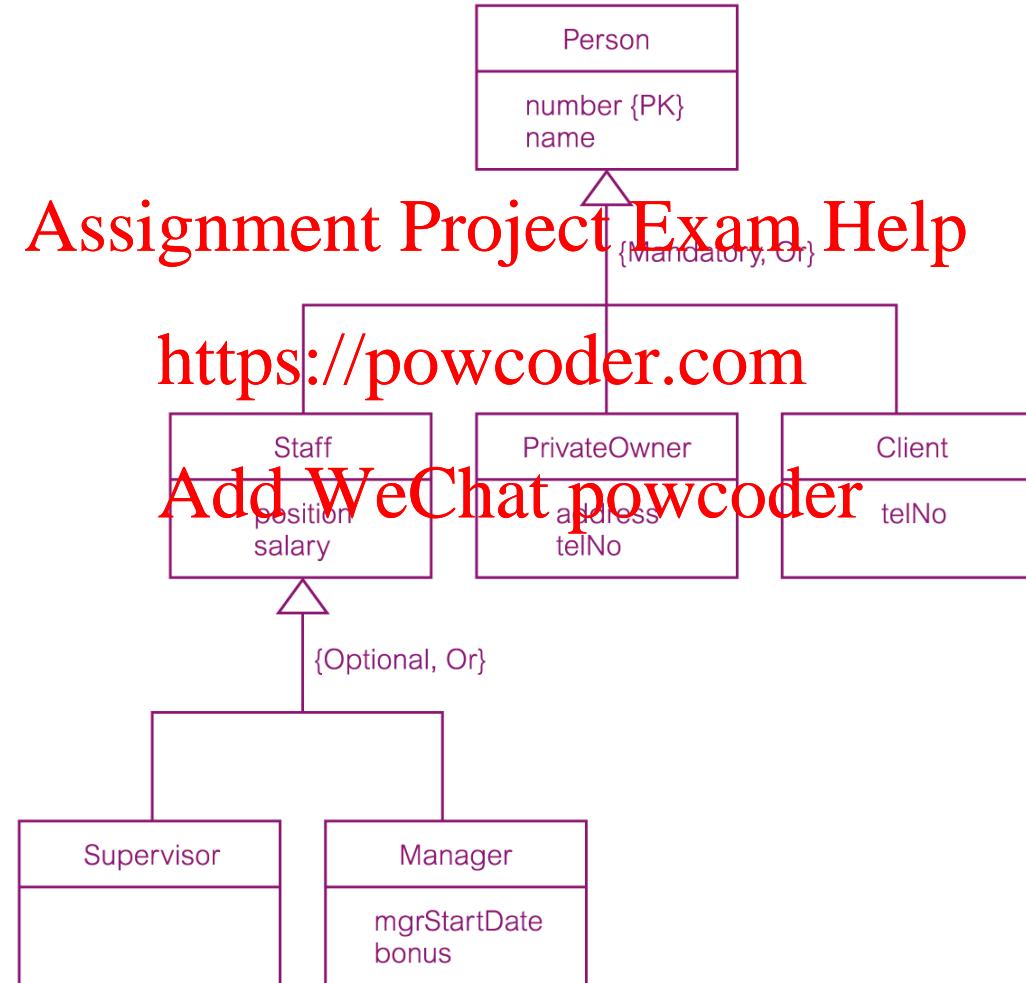
# DreamHome Worked Example - Staff Superclass with Supervisor and Manager Subclasses



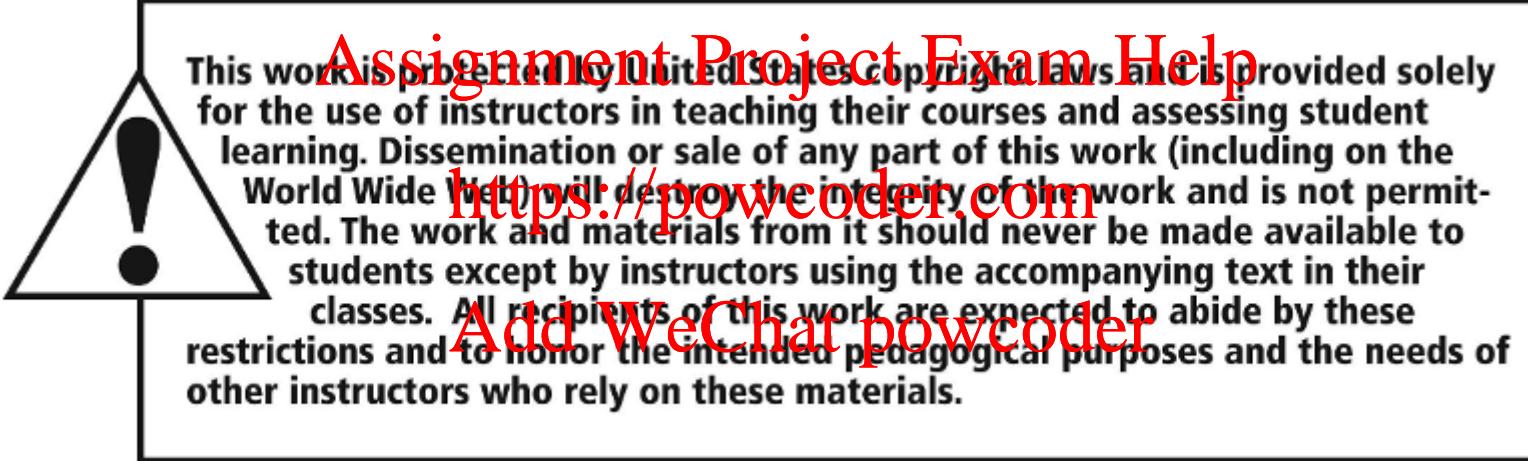
# DreamHome Worked Example - Owner Superclass with PrivateOwner and BusinessOwner Subclasses



# DreamHome Worked Example - Person Superclass with Staff, PrivateOwner, and Client Subclasses



# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 14

<https://powcoder.com> Normalization

Add WeChat powcoder

# Learning Objectives (1 of 3)

- 14.1** The purpose of normalization.
- 14.2** How normalization can be used when designing a relational database.  
*Assignment Project Exam Help*
- 14.3** The potential problems associated with redundant data in base relations.  
*Add WeChat powcoder*
- 14.4** The concept of functional dependency, which describes the relationship between attributes.
- 14.5** The characteristics of functional dependencies used in normalization.

# Learning Objectives (2 of 3)

**14.6** How to identify functional dependencies for a given relation.

**14.7** How functional dependencies identify the primary key for a relation. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com>

**14.8** How to undertake the process of normalization.

[Add WeChat powcoder](#)

**14.9** How normalization uses functional dependencies to group attributes into relations that are in a known normal form.

# Learning Objectives (3 of 3)

**14.10** How to identify the most commonly used normal forms, namely First Normal Form (1NF), Second Normal Form (2NF), and Third Normal Form (3NF).

**14.11** The problems associated with relations that break the rules of 1NF, 2NF, or 3NF.

**14.12** How to represent attributes shown on a form as 3NF relations using normalization.

# Purpose of Normalization (1 of 3)

- Normalization is a technique for producing a set of suitable relations that support the data requirements of an enterprise

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Purpose of Normalization (2 of 3)

- Characteristics of a suitable set of relations include:
  - the **minimal** number of attributes necessary to support ~~Assignment Project Exam Help~~ <https://powcoder.com>
  - attributes with a close logical relationship are found in the same relation;
  - **minimal** redundancy with each attribute represented only once with the important exception of attributes that form all or part of foreign keys.

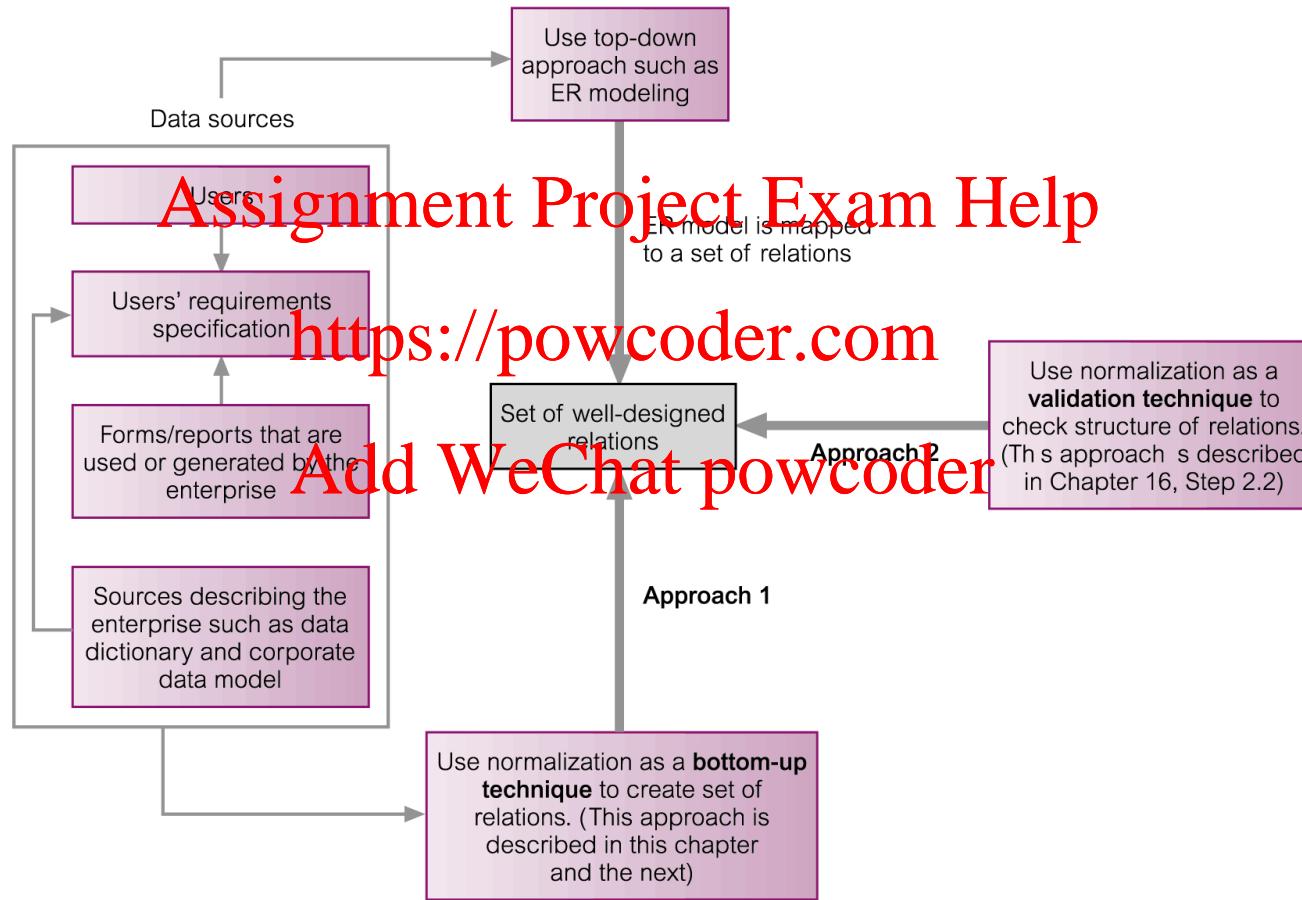
# Purpose of Normalization (3 of 3)

- The benefits of using a database that has a suitable set of relations is that the database will be:
  - easier for the user to access and maintain the data;
  - take up minimal storage space on the computer.

<https://powcoder.com>

Add WeChat powcoder

# How Normalization Supports Database Design



# Data Redundancy and Update Anomalies (1 of 8)

- Major aim of relational database design is to group attributes into relations to minimize data redundancy.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Data Redundancy and Update Anomalies (2 of 8)

- Potential benefits for implemented database include:
  - Updates to the data stored in the database are achieved with a minimal number of operations thus reducing the opportunities for data inconsistencies.
  - Reduction in the file storage space required by the base relations thus minimizing costs

# Data Redundancy and Update Anomalies (3 of 8)

- Problems associated with data redundancy are illustrated by comparing the Staff and Branch relations with the Staff Branch relation

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Data Redundancy and Update Anomalies (4 of 8)

## Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

# Data Redundancy and Update Anomalies (5 of 8)

## Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Data Redundancy and Update Anomalies (6 of 8)

## StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

# Data Redundancy and Update Anomalies (7 of 8)

- StaffBranch relation has redundant data; the details of a branch are repeated for every member of staff.
- In contrast, the branch information appears only once for each branch in the Branch relation and only the branch number (branchNo) is repeated in the Staff relation, to represent where each member of staff is located.  
Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder

# Data Redundancy and Update Anomalies (8 of 8)

- Relations that contain redundant information may potentially suffer from update anomalies.
- Types of update anomalies include
  - Insertion <https://powcoder.com>
  - Deletion
  - Modification

# Lossless-Join and Dependency Preservation Properties

- Two important properties of decomposition.
  - **Lossless-join property** enables us to find any instance of the original relation from corresponding instances in the smaller relations.
  - **Dependency preservation property** enables us to enforce a constraint on the original relation by enforcing some constraint on each of the smaller relations.

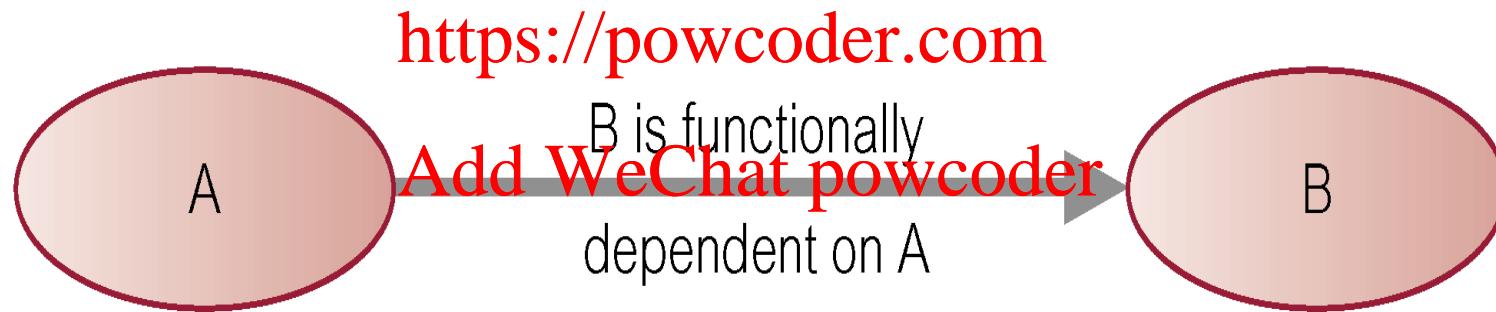
# Functional Dependencies

- Important concept associated with normalization.
- Functional dependency describes relationship between attributes. **Assignment Project Exam Help**
- For example, if A and B are attributes of relation R, B is functionally dependent on A (denoted  $A \rightarrow B$ ), if each value of A in R is associated with exactly one value of B in R.

# Characteristics of Functional Dependencies (1 of 4)

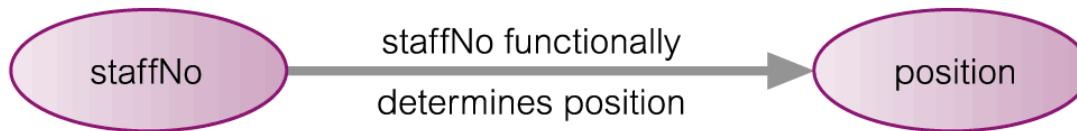
- Property of the meaning or semantics of the attributes in a relation.
- Diagrammatic representation.

Assignment Project Exam Help



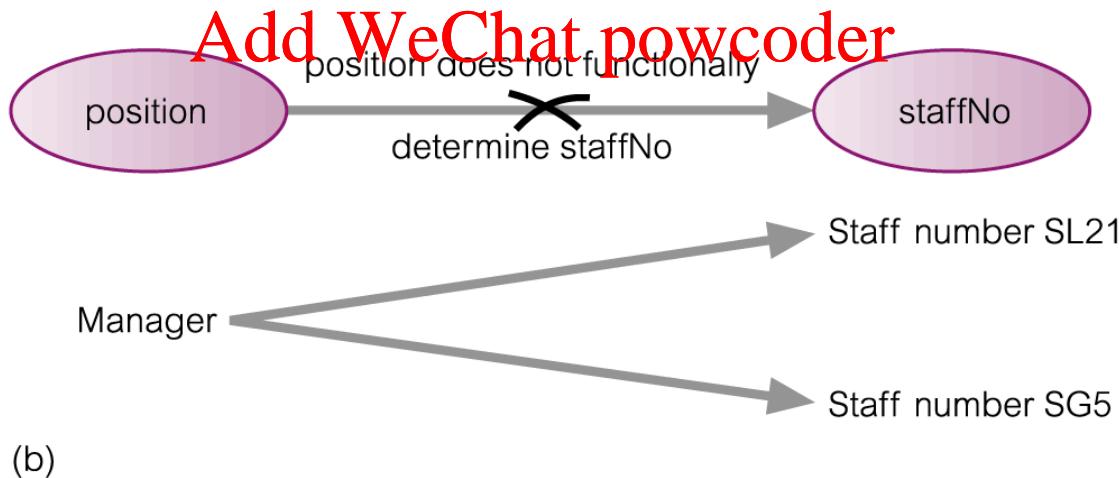
- The **determinant** of a functional dependency refers to the attribute or group of attributes on the left-hand side of the arrow.

# An Example Functional Dependency



Assignment Project Exam Help

Staff number SL21 → Manager  
(a) <https://powcoder.com>



# Example Functional Dependency That Holds for All Time (1 of 2)

- Consider the values shown in staffNo and sName attributes of the Staff relation (see Slide 12).
- Based on sample data, the following functional dependencies appear to hold.  
 $\text{staffNo} \rightarrow \text{sName}$   
 $\text{sName} \rightarrow \text{staffNo}$

# Example Functional Dependency That Holds for All Time (2 of 2)

- However, the only functional dependency that remains true for all possible values for the staffNo and sName attributes of the Staff relation is:

staffNo → sName

<https://powcoder.com>

Add WeChat powcoder

## Characteristics of Functional Dependencies (2 of 4)

- Determinants should have the minimal number of attributes necessary to maintain the functional dependency with the attribute(s) on the right hand-side.
- This requirement is called **full functional dependency**.  
<https://powcoder.com>

Add WeChat powcoder

## Characteristics of Functional Dependencies (3 of 4)

- Full functional dependency indicates that if A and B are attributes of a relation, B is fully functionally dependent on A, if B is functionally dependent on A, but not on any proper subset of A.

[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

# Example Full Functional Dependency

- Exists in the Staff relation (see Slide 12).

$\text{staffNo}, \text{sName} \rightarrow \text{branchNo}$

Assignment Project Exam Help

- True - each value of  $(\text{staffNo}, \text{sName})$  is associated with a single value of  $\text{branchNo}$ .
- However,  $\text{branchNo}$  is also functionally dependent on a subset of  $(\text{staffNo}, \text{sName})$ , namely  $\text{staffNo}$ . Example above is a **partial dependency**.

## Characteristics of Functional Dependencies (4 of 4)

- Main characteristics of functional dependencies used in normalization:
  - There is a one-to-one relationship between the attribute(s) on the left-hand side (determinant) and those on the right-hand side of a functional dependency.
  - Holds for **all** time.
  - The determinant has the **minimal** number of attributes necessary to maintain the dependency with the attribute(s) on the right hand-side.

# Transitive Dependencies

- Important to recognize a transitive dependency because its existence in a relation can potentially cause update anomalies. **Assignment Project Exam Help**
- Transitive dependency describes a condition where A, B, and C are attributes of a relation such that if  $A \rightarrow B$  and  $B \rightarrow C$ , then C is transitively dependent on A via B (provided that A is not functionally dependent on B or C).  
<https://powcoder.com>  
Add WeChat powcoder

# Example Transitive Dependency

- Consider functional dependencies in the StaffBranch relation (see Slide 12).

StaffNo → sName, position, salary, branchNo, bAddress

branchNo → bAddress

- Transitive dependency, branchNo → bAddress exists on staffNo via branchNo.

# The Process of Normalization (1 of 4)

- Formal technique for analyzing a relation based on its primary key and the functional dependencies between the attributes of that relation
- Often executed as a series of steps. Each step corresponds to a specific normal form, which has known properties.

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

# Identifying Functional Dependencies (1 of 2)

- Identifying all functional dependencies between a set of attributes is relatively simple if the meaning of each attribute and the relationships between the attributes are well understood.
- This information should be provided by the enterprise in the form of discussions with users and/or documentation such as the users' requirements specification.  
<https://powcoder.com>  
Add WeChat powcoder

# Identifying Functional Dependencies (2 of 2)

- However, if the users are unavailable for consultation and/or the documentation is incomplete then depending on the database application it may be necessary for the database designer to use their common sense and/or experience to provide <https://powcoder.com> information.

Add WeChat powcoder

# Example - Identifying a Set of Functional Dependencies for the StaffBranch Relation (1 of 2)

- Examine semantics of attributes in StaffBranch relation (see Slide 12). Assume that position held and branch determine a member of staff's salary

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

## Example - Identifying a Set of Functional Dependencies for the StaffBranch Relation (2 of 2)

- With sufficient information available, identify the functional dependencies for the StaffBranch relation as:  
 $\text{staffNo} \rightarrow \text{sName, position, salary, branchNo, bAddress}$   
 $\text{branchNo} \rightarrow \text{bAddress}$   
 $\text{bAddress} \rightarrow \text{branchNo}$   
 $\text{branchNo, position} \rightarrow \text{salary}$   
 $\text{bAddress, position} \rightarrow \text{salary}$

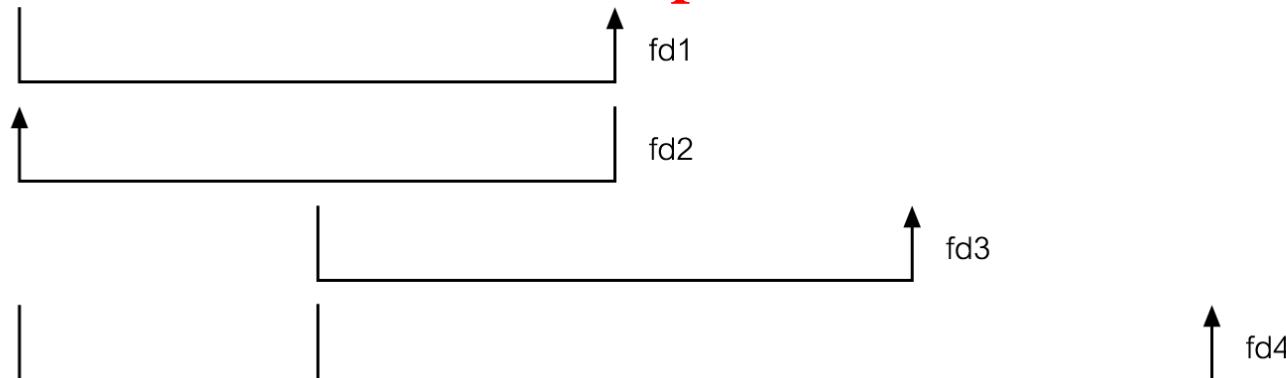
# Example - Using Sample Data to Identify Functional Dependencies (1 of 3)

- Consider the data for attributes denoted A, B, C, D, and E in the Sample relation (see Slide 33).
- Important to establish that sample data values shown in relation are representative of all possible values that can be held by attributes A, B, C, D, and E. Assume true despite the relatively small amount of data shown in this relation.

# Example - Using Sample Data to Identify Functional Dependencies (2 of 3)

Sample Relation

A	B	C	D	E
a	b	z	w	q
e	b	r	w	p
a	d	z	w	t
e	c	r	w	q
a	f	z	s	t
e	c	r	s	t



# Example - Using Sample Data to Identify Functional Dependencies (3 of 3)

- Function dependencies between attributes A to E in the Sample relation.

	Assignment	Project	Exam	Help
$A \rightarrow C$			(fd1)	
$C \rightarrow A$			(fd2)	<a href="https://powcoder.com">https://powcoder.com</a>
$B \rightarrow D$		Add WeChat	(fd3)	powcoder
$A, B \rightarrow E$			(fd4)	

# Identifying the Primary Key for a Relation Using Functional Dependencies

- Main purpose of identifying a set of functional dependencies for a relation is to specify the set of integrity constraints that must hold on a relation.
- An important integrity constraint to consider first is the identification of candidate keys, one of which is selected to be the primary key for the relation.

# Example - Identify Primary Key for StaffBranch Relation (1 of 2)

- StaffBranch relation has five functional dependencies (see Slide 31).
- The determinants are staffNo, branchNo, bAddress, (branchNo, position), and (bAddress, position).  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)  
[Add WeChat powcoder](#)
- To identify all candidate key(s), identify the attribute (or group of attributes) that uniquely identifies each tuple in this relation.

## Example - Identify Primary Key for StaffBranch Relation (2 of 2)

- All attributes that are not part of a candidate key should be functionally dependent on the key.
- The only candidate key and therefore primary key for StaffBranch relation, is staffNo, as all other attributes of the relation are functionally dependent on staffNo.

Add WeChat powcoder

# Example - Identifying Primary Key for Sample Relation

- Sample relation has four functional dependencies (see Slide 31).
- The determinants in the Sample relation are A, B, C, and (A, B). However, the only determinant that functionally determines all the other attributes of the relation is (A, B).
- (A, B) is identified as the primary key for this relation.

# The Process of Normalization (2 of 4)

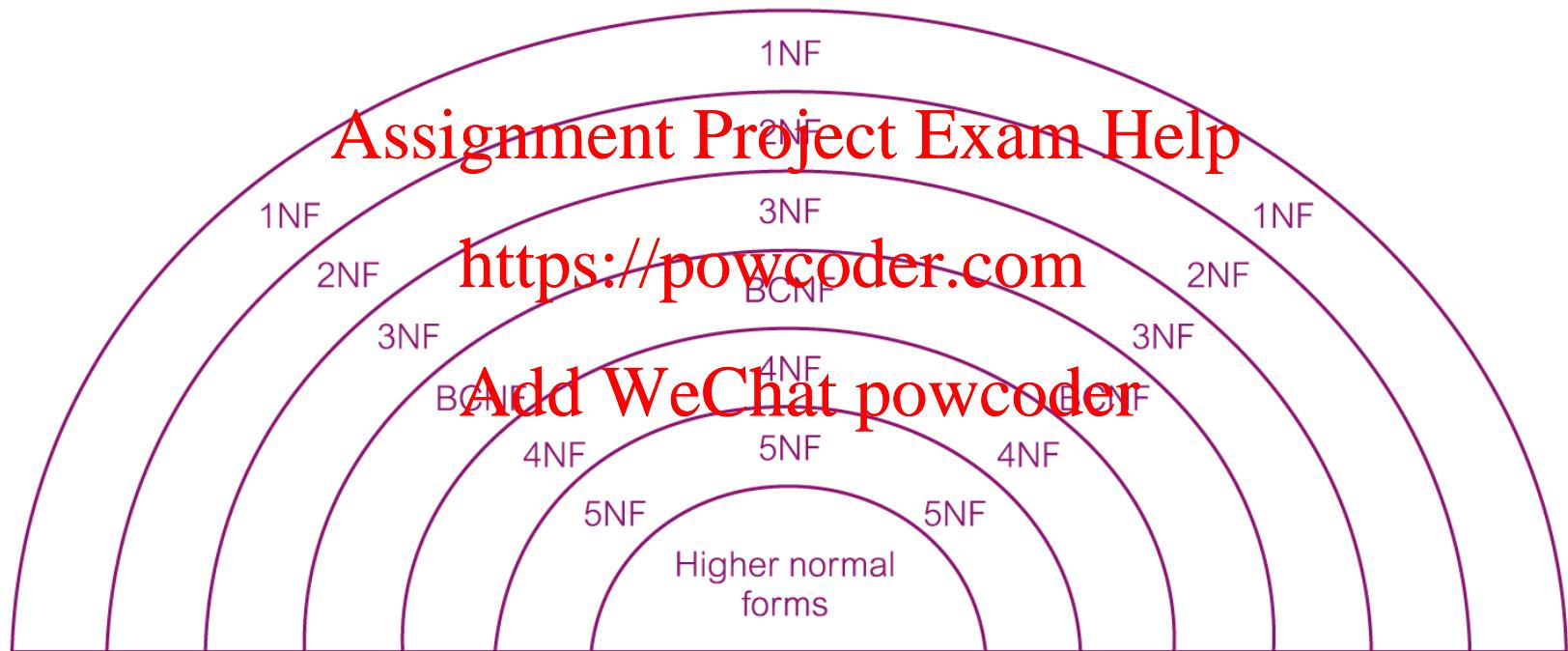
- As normalization proceeds, the relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies

**Assignment Project Exam Help**

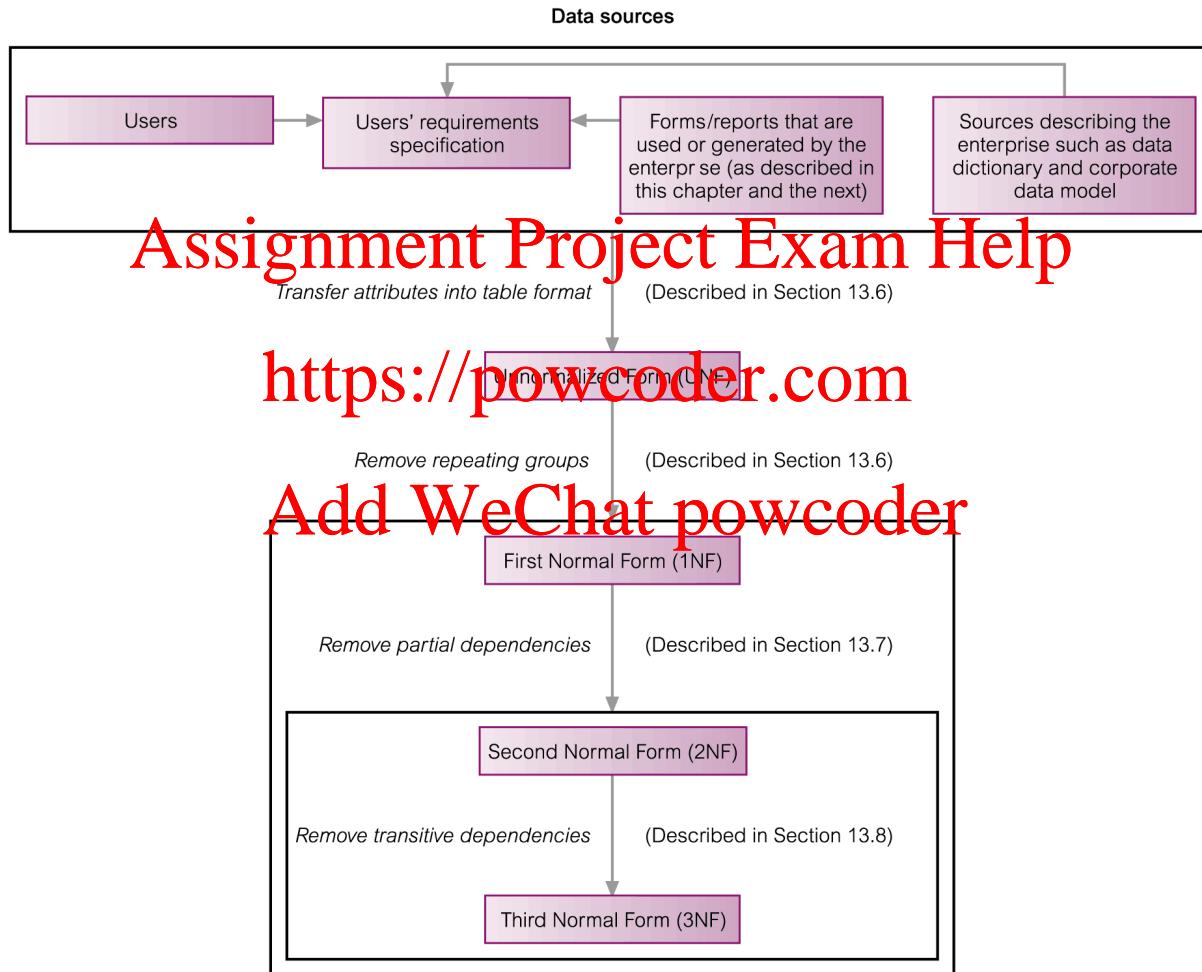
<https://powcoder.com>

Add WeChat powcoder

# The Process of Normalization (3 of 4)



# The Process of Normalization (4 of 4)



# Unnormalized Form (UNF)

- A table that contains one or more repeating groups.
- To create an unnormalized table
  - Transform the data from the information source (e.g. form) into table format with columns and rows.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# First Normal Form (1NF)

- A relation in which the intersection of each row and column contains one and only one value.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## UNF to 1NF (1 of 2)

- Nominate an attribute or group of attributes to act as the key for the unnormalized table.
- Identify the repeating group(s) in the unnormalized table which repeats for the key attribute(s).  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

## UNF to 1NF (2 of 2)

- Remove the repeating group by
  - Entering appropriate data into the empty columns of rows containing the repeating data ('Flattening' the table).
  - Or by <https://powcoder.com>
  - Placing the repeating data along with a copy of the original key attribute(s) into a separate relation.

# Second Normal Form (2NF) (1 of 2)

- Based on the concept of full functional dependency.
- Full functional dependency indicates that if
  - A and B are attributes of a relation,
  - B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.

## Second Normal Form (2NF) (2 of 2)

- A relation that is in 1NF and every non-primary-key attribute is fully functionally dependent on the primary key.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# 1NF to 2NF

- Identify the primary key for the 1NF relation.
- Identify the functional dependencies in the relation.  
**Assignment Project Exam Help**
- If partial dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their determinant.  
**Add WeChat powcoder**

## Third Normal Form (3NF) (1 of 2)

- Based on the concept of transitive dependency.
- Transitive Dependency is a condition where
  - A, B and C are attributes of a relation such that if  $A \rightarrow B$  and  $B \rightarrow C$ , then C is transitively dependent on A through B.  
<https://powcoder.com>
  - then C is transitively dependent on A through B.  
(Provided that A is not functionally dependent on B or C).

## Third Normal Form (3NF) (2 of 2)

- A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on the primary key

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## 2NF to 3NF

- Identify the primary key in the 2NF relation.
- Identify functional dependencies in the relation.
- If transitive dependencies exist on the primary key remove them by placing them in a new relation along with a copy of their dominant.

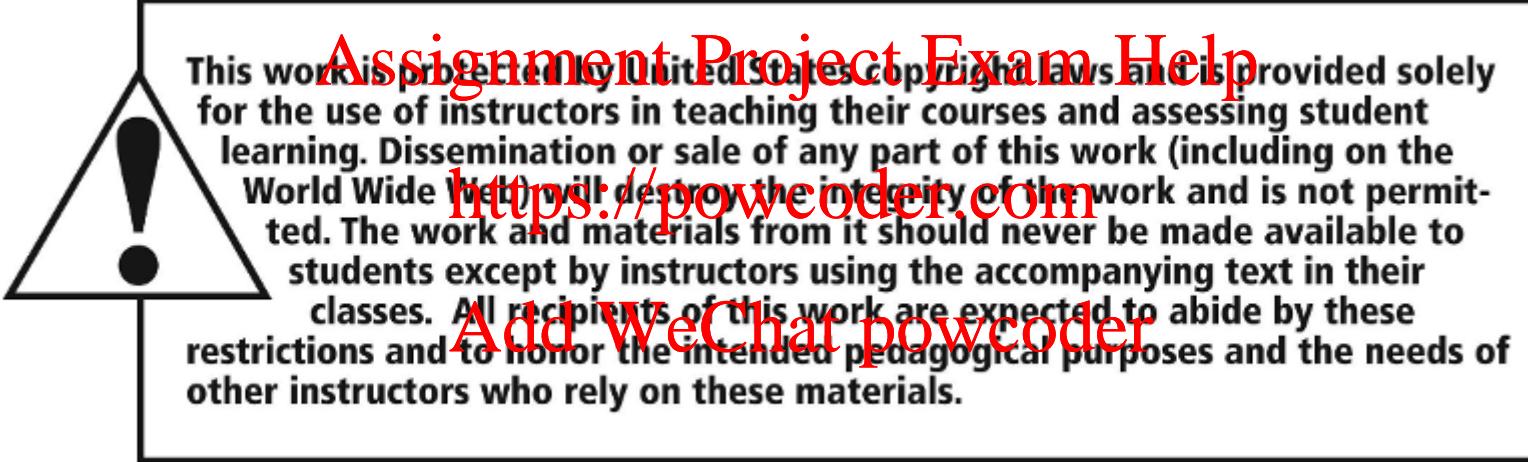
Assignment Project Exam Help

Add WeChat powcoder

# General Definitions of 2NF and 3NF

- Second normal form (2NF)
  - A relation that is in first normal form and every non-primary key attribute is fully functionally dependent on **any candidate key**.  
<https://powcoder.com>
- Third normal form (3NF)
  - A relation that is in first and second normal form and in which no non-primary-key attribute is transitively dependent on **any candidate key**.

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 15

<https://powcoder.com> Advanced Normalization

Add WeChat powcoder

# Learning Objectives (1 of 4)

**15.1** How inference rules can identify a set of **all** functional dependencies for a relation.

**15.2** How Inference rules called Armstrong's Axioms can identify a **minimal** set of useful functional dependencies from the set of all functional dependencies for a relation.

Add WeChat powcoder  
<https://powcoder.com>

# Learning Objectives (2 of 4)

**15.3** Normal forms that go beyond Third Normal Form (3NF), which includes Boyce-Codd Normal Form (BCNF), Fourth Normal Form (4NF), and Fifth Normal Form (5NF).

**15.4** How to identify Boyce-Codd Normal Form (BCNF).  
<https://powcoder.com>

**15.5** How to represent attributes shown on a report as BCNF relations using normalization.  
[Add WeChat powcoder](#)

# Learning Objectives (3 of 4)

**15.6** Concept of multi-valued dependencies and Fourth Normal Form (4NF).

**15.7** The problems associated with relations that break the rules of 4NF.

<https://powcoder.com>

**15.8** How to create 4NF relations from a relation, which breaks the rules of 4NF.

[Add WeChat powcoder](#)

# Learning Objectives (4 of 4)

**15.9** Concept of join dependency and Fifth Normal Form (5NF).

**15.10** The problems associated with relations that break the rules of 5NF. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com>

**15.11** How to create 5NF relations from a relation, which breaks the rules of 5NF. [Add WeChat powcoder](#)

# More on Functional Dependencies

- The complete set of functional dependencies for a given relation can be very large.
- Important to find an approach that can reduce the set to a manageable size  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

# Inference Rules for Functional Dependencies (1 of 5)

- Need to identify a set of functional dependencies (represented as X) for a relation that is smaller than the complete set of functional dependencies (represented as Y) for that relation and has the property that every functional dependency in Y is implied by the functional dependencies in X.

Add WeChat powcoder

# Inference Rules for Functional Dependencies (2 of 5)

- The set of all functional dependencies that are implied by a given set of functional dependencies X is called the **closure of X**, written  $X^+$ .
- A set of inference rules, called **Armstrong's axioms**, specifies how new functional dependencies can be inferred from given ones.

# Inference Rules for Functional Dependencies (3 of 5)

- Let A, B, and C be subsets of the attributes of the relation R. Armstrong's axioms are as follows:

## 1. Reflexivity

If B is a subset of A, then  $A \rightarrow B$

## 2. Augmentation

If  $A \rightarrow B$ , then  $A,C \rightarrow B,C$

## 3. Transitivity

If  $A \rightarrow B$  and  $B \rightarrow C$ , then  $A \rightarrow C$

# Inference Rules for Functional Dependencies (4 of 5)

- Further rules can be derived from the first three rules that simplify the practical task of computing  $X^+$ . Let D be another subset of the attributes of relation R, then:

## 4. Self-determination

<https://powcoder.com>

$$A \rightarrow A$$

## 5. Decomposition

Add WeChat powcoder

If  $A \rightarrow BC$ , then  $A \rightarrow B$  and  $A \rightarrow C$

# Inference Rules for Functional Dependencies (5 of 5)

## 6. Union

If  $A \rightarrow B$  and  $A \rightarrow C$ , then  $A \rightarrow B,C$

## Assignment Project Exam Help 7. Composition

If  $A \rightarrow B$  and  $C \rightarrow D$  then  $A,C \rightarrow B,D$

Add WeChat powcoder

# Minimal Sets of Functional Dependencies (1 of 2)

- A set of functional dependencies Y is covered by a set of functional dependencies X, if every functional dependency in Y is also in X; that is, every dependency in Y can be inferred from X.  
<https://powcoder.com>
- A set of functional dependencies X is minimal if it satisfies the following conditions:
  - Every dependency in X has a single attribute on its right-hand side.

## Minimal Sets of Functional Dependencies (2 of 2)

- We cannot replace any dependency  $A \rightarrow B$  in  $X$  with dependency  $C \rightarrow B$ , where  $C$  is a proper subset of  $A$ , and still have a set of dependencies that is equivalent to  $X$ .
- We cannot remove any dependency from  $X$  and still have a set of dependencies that is equivalent to  $X$ .

# Boyce–Codd Normal Form (BCNF) (1 of 3)

- Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with the general definition of 3NF.  
<https://powcoder.com>
- Boyce–Codd normal form (BCNF)
  - A relation is BCNF if and only if every determinant is a candidate key.

# Boyce–Codd Normal Form (BCNF) (2 of 3)

- Difference between 3NF and BCNF is that for a functional dependency  $A \rightarrow B$ , 3NF allows this dependency in a relation if B is a primary-key attribute and A is not a candidate key. Whereas, BCNF insists that for this dependency to remain in a relation, A must be a candidate key.
- Every relation in BCNF is also in 3NF. However, a relation in 3NF is not necessarily in BCNF.

Assignment Project Exam Help

Add WeChat powcoder

# Boyce–Codd Normal Form (BCNF) (3 of 3)

- Violation of BCNF is quite rare.
- The potential to violate BCNF may occur in a relation that:
  - contains two (or more) composite candidate keys;
  - the candidate keys overlap, that is have at least one attribute in common.

# Review of Normalization (UNF to BCNF) (1 of 5)

*DreamHome*  
Property Inspection Report

*DreamHome*  
Property Inspection Report

**Assignment Project Exam Help**

Property Number PG4

Property Address 10 Lakewood St, Glasgow

Add WeChat powcoder

Inspection Date	Inspection Time	Comments	Staff no	Staff Name	Car Registration
18-Oct-12	10.00	Need to replace crockery	SG37	Ann Beech	M231 JGR
22-Apr-13	09.00	In good order	SG14	David Ford	M533 HDR
1-Oct-13	12.00	Damp rot in bathroom	SG14	David Ford	N721 HFR

Page 1

# Review of Normalization (UNF to BCNF) (2 of 5)

## StaffPropertyInspection

propertyNo	pAddress	iDate	iTime	Comments	staffNo	sName	carReg
PG4	6 Lawrence St, Glasgow	18-Oct-12 22-Apr-13 1-Oct-13	10.00 09.00 12.00	Need to replace crockery In good order Damp rot in bathroom	SG37 SG14 SG14	Ann Beech David Ford David Ford	M231 JGR M533 HDR N721 HFR
PG16	5 Novar Dr, Glasgow	22-Apr-13 24-Oct-13	13.00 14.00	Replace living room carpet Good condition	SG14 SG37	David Ford Ann Beech	M533 HDR N721 HFR

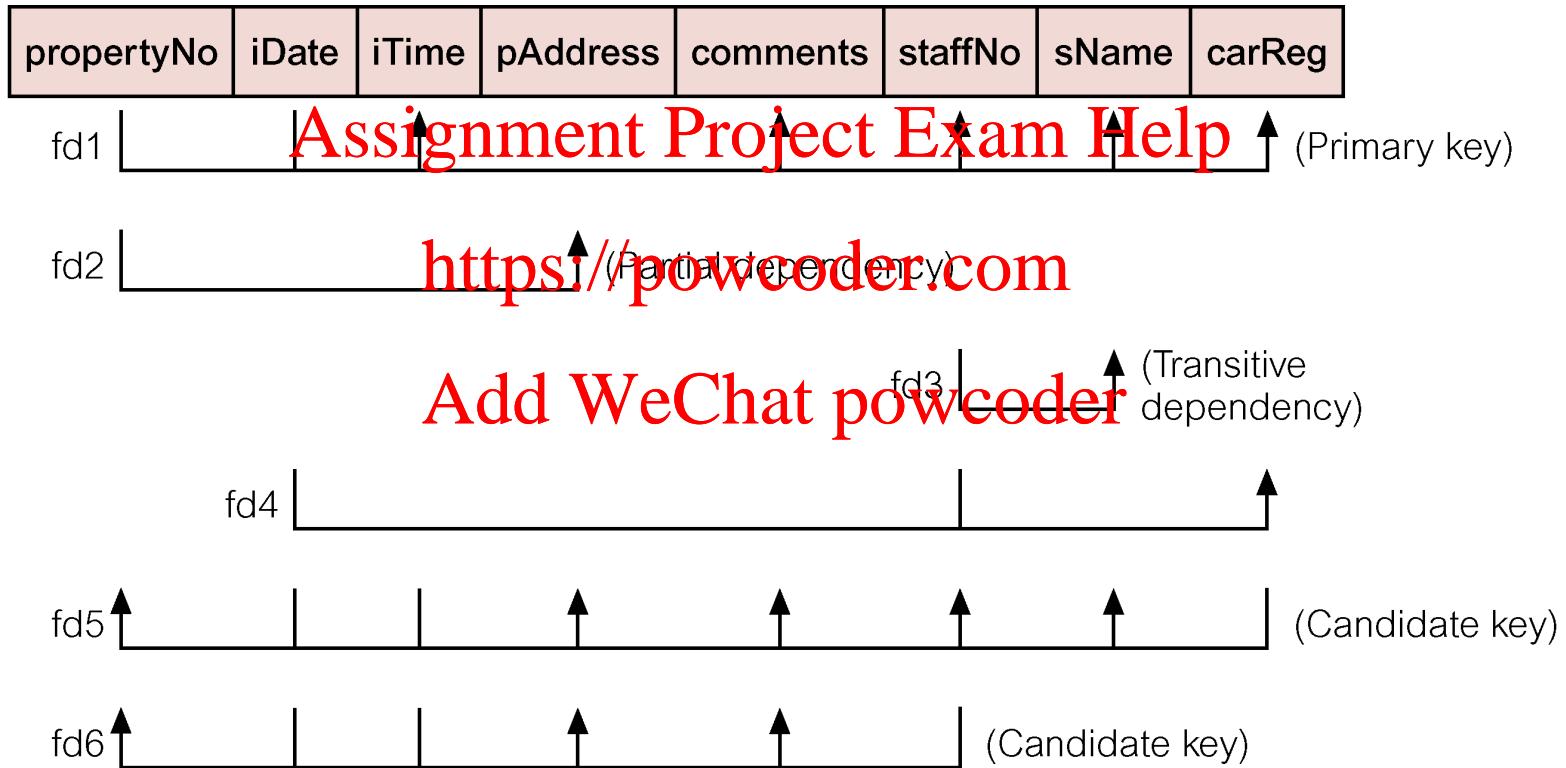
# Review of Normalization (UNF to BCNF) (3 of 5)

## StaffPropertyInspection

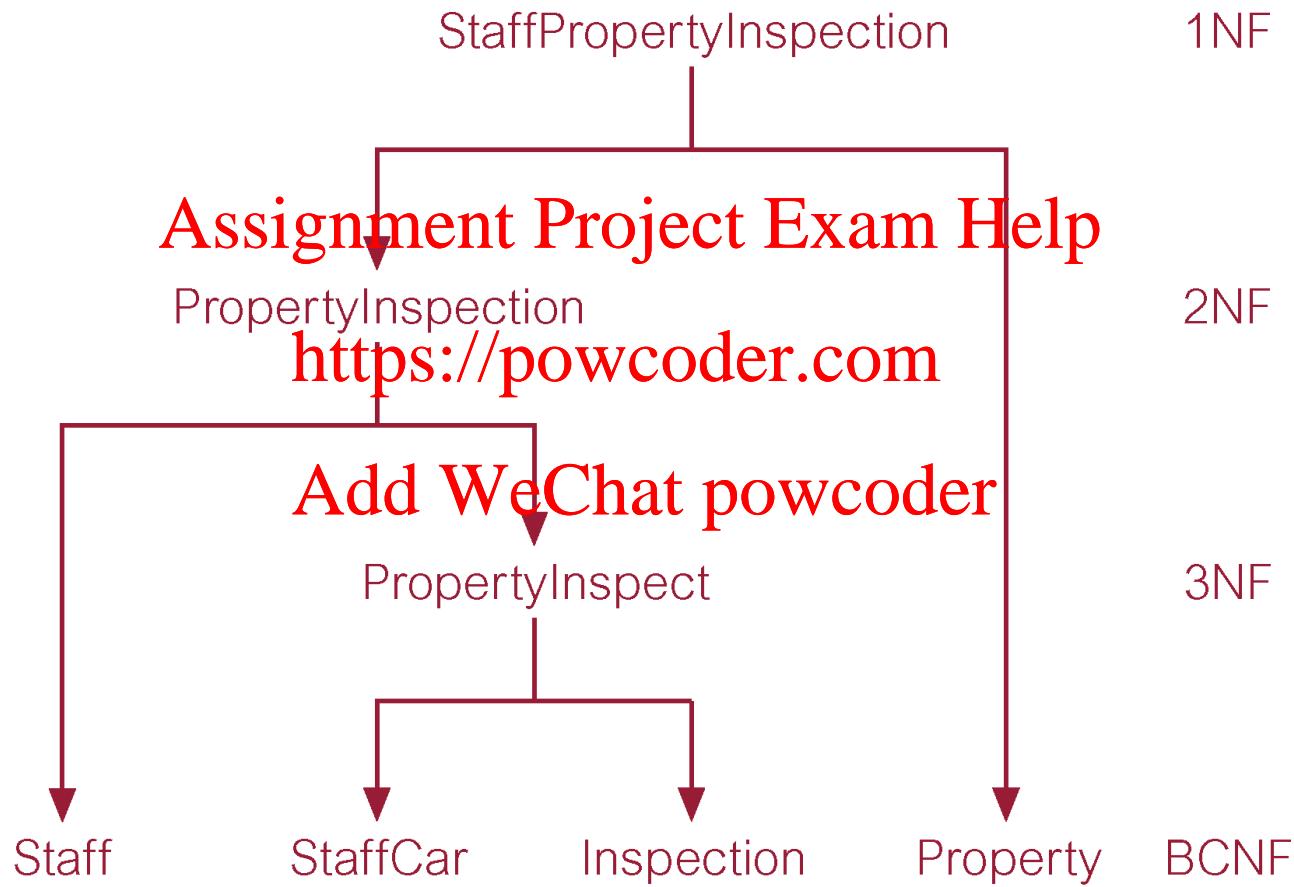
propertyNo	iDate	iTime	pAddress	Comments	staffNo	sName	carReg
PG4	18-Oct-12	10.00	6 Lawrence St, Glasgow	Need to replace crockery	SG37	Ann Beech	M231 JGR
PG4	22-Apr-13	09.00	6 Lawrence St, Glasgow	In good order	SG14	David Ford	M533 HDR
PG4	1-Oct-13	12.00	6 Lawrence St, Glasgow	Damp spot in bathroom	SG14	David Ford	N721 HFR
PG16	22-Apr-13	13.00	5 Novar Dr, Glasgow	Replace living room carpet	SG14	David Ford	M533 HDR
PG16	24-Oct-13	14.00	5 Novar Dr, Glasgow	Good condition	SG37	Ann Beech	N721 HFR

# Review of Normalization (UNF to BCNF) (4 of 5)

StaffPropertyInspection



# Review of Normalization (UNF to BCNF) (5 of 5)



# Fourth Normal Form (4NF) (1 of 5)

- Although BCNF removes anomalies due to functional dependencies, another type of dependency called a multi-valued dependency (MVD) can also cause data redundancy.
- Possible existence of multi-valued dependencies in a relation is due to 1NF and can result in data redundancy.

# Fourth Normal Form (4NF) (2 of 5)

- Multi-valued Dependency (MVD)
  - Dependency between attributes (for example, A, B, and C) in a relation, such that for each value of A there is a set of values for B and a set of values for C. However, the set of values for B and C are independent of each other.

Add WeChat powcoder

## Fourth Normal Form (4NF) (3 of 5)

- MVD between attributes A, B, and C in a relation using the following notation:

A- >> B [Assignment](#) [Project](#) [Exam](#) [Help](#)

A- >> C <https://powcoder.com>

[Add WeChat](#) [powcoder](#)

## Fourth Normal Form (4NF) (4 of 5)

- A multi-valued dependency can be further defined as being trivial or nontrivial.

A MVD ~~Assignment Project Exam Help~~ is defined as being trivial if (a) B is a subset of A **or** (b)  $A \cup B = R$ .

A MVD is defined as being nontrivial if neither (a) nor (b) are satisfied.  
<https://powcoder.com>  
Add WeChat powcoder

A trivial MVD does not specify a constraint on a relation, while a nontrivial MVD does specify a constraint.

# Fourth Normal Form (4NF) (5 of 5)

- Defined as a relation that is in Boyce-Codd Normal Form and contains no nontrivial multi-valued dependencies.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# 4NF - Example

**BranchStaffOwner**

branchNo	sName	oName
B003	Ann Beech	Carol Farrel
B003	David Ford	Carol Farrel
B003	Ann Beech	Tina Murphy
B003	David Ford	Tina Murphy

Add WeChat powcoder



**BranchStaff**

branchNo	sName
B003	Ann Beech
B003	David Ford

**BranchOwner**

branchNo	oName
B003	Carol Farrel
B003	Tina Murphy

## Fifth Normal Form (5NF) (1 of 2)

- A relation decompose into two relations must have the lossless-join property, which ensures that no spurious tuples are generated when relations are reunited through a natural join operation.  
<https://powcoder.com>
- However, there are requirements to decompose a relation into more than two relations. Although rare, these cases are managed by join dependency and fifth normal form (5NF).  
[Add WeChat powcoder](#)

## Fifth Normal Form (5NF) (2 of 2)

- Defined as a relation that has no join dependency.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# 5NF – Example (1 of 2)

(a) PropertyItemSupplier (Illegal state)

propertyNo	itemDescription	supplierNo
PG4	Bed	S1
PG4	Chair	S2
PG16	Bed	S2

Assignment Project Exam Help

<https://powcoder.com>

(b) PropertyItemSupplier (Legal state)

propertyNo	itemDescription	supplierNo
PG4	Bed	S1
PG4	Chair	S2
PG16	Bed	S2
PG4	Bed	S2

When this tuple is added to relation.

This new tuple must also be added to exist in any legal state of the relation.

## 5NF – Example (2 of 2)

### PropertyNo

propertyNo	itemDescription
PG4	Bed
PG4	Chair
PG16	Bed

Assignment Project Exam Help

<https://powcoder.com>

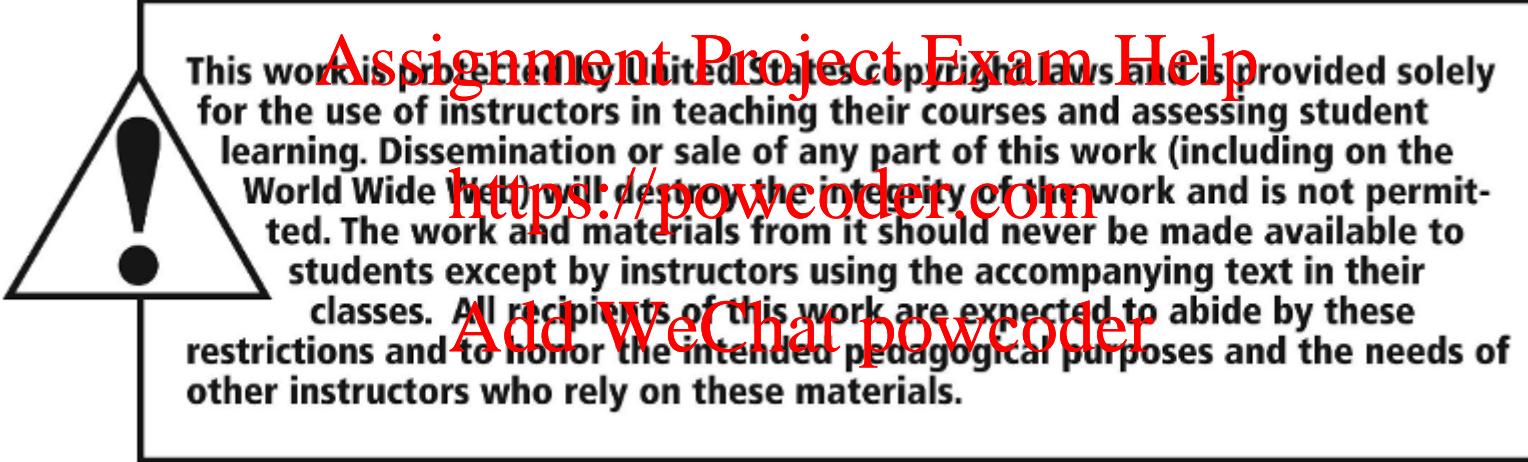
### ItemSupplier

Add WeChat powcoder

itemDescription	supplierNo
Bed	S1
Chair	S2
Bed	S2

propertyNo	supplierNo
PG4	S1
PG4	S2
PG16	S2

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 16

<https://powcoder.com>  
Methodology Conceptual

Databases Design  
Add WeChat powcoder

# Learning Objectives (1 to 3)

**16.1** The purpose of a design methodology.

**16.2** Database design has three main phases: conceptual, logical, and physical design.  
*Assignment Project Exam Help*

**16.3** How to decompose the scope of the design into specific views of the enterprise.

*Add WeChat powcoder*

# Learning Objectives (2 to 3)

**16.4** How to use Entity–Relationship (ER) modeling to build a conceptual data model based on the data requirements of an enterprise

**Assignment Project Exam Help**

**16.5** How to validate the resultant conceptual model to ensure it is a true and accurate representation of the data requirements enterprise.

<https://powcoder.com>

Add WeChat powcoder

# Learning Objectives (3 to 3)

**16.6** How to document the process of conceptual database design.

**16.7** End-users play an integral role throughout the process of conceptual database design.

<https://powcoder.com>

Add WeChat powcoder

# Design Methodology

- A structured approach that uses procedures, techniques, tools, and documentation aids to support and facilitate the process of design.

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

# Database Design Methodology

- Three main phases
  - Conceptual database design
  - Logical database design
  - Physical database design

Add WeChat powcoder

# Conceptual Database Design

- The process of constructing a model of the data used in an enterprise, independent of **all** physical considerations.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Logical Database Design

- The process of constructing a model of the data used in an enterprise based on a specific data model (e.g. relational), **Assignment Project Exam Help** and other physical considerations.

<https://powcoder.com>

Add WeChat powcoder

# Physical Database Design

- The process of producing a description of the implementation of the database on secondary storage; it describes the base relations, file organizations, and indexes design used to achieve efficient access to the data, and any associated integrity constraints and security measures.

Add WeChat powcoder

# Critical Success Factors in Database Design (1 to 2)

- Work interactively with the users as much as possible.
- Follow a structured methodology throughout the data modeling process.  
**Assignment Project Exam Help**
- Employ a data-driven approach.  
**https://powcoder.com**
- Incorporate structural and integrity considerations into the data models.  
**Add WeChat powcoder**
- Combine conceptualization, normalization, and transaction validation techniques into the data modeling methodology.

# Critical Success Factors in Database Design (2 to 2)

- Use diagrams to represent as much of the data models as possible.
- Use a Database Design Language (DBDL) to represent additional data semantics.  
<https://powcoder.com>
- Build a data dictionary to supplement the data model diagrams.
- Be willing to repeat steps.

# Overview Database Design Methodology (1 to 7)

## Conceptual database design

- Step 1 Build conceptual data model
  - Step 1.1 Identify entity types
  - Step 1.2 Identify relationship types
  - Step 1.3 Identify and associate attributes with entity or relationship types
  - Step 1.4 Determine attribute domains
  - Step 1.5 Determine candidate, primary, and alternate key attributes

# Overview Database Design Methodology (2 to 7)

- Step 1.6 Consider use of enhanced modeling concepts (optional step)
- Step 1.7 ~~Assignment Project Exam Help~~
- Step 1.8 Validate conceptual model against user transactions <https://powcoder.com>
- Step 1.9 Review conceptual data model with user ~~Add WeChat powcoder~~

# Overview Database Design Methodology (3 to 7)

Logical database design for the relational model

- Step 2 Build and validate logical data model
  - Step 2.1 Derive relations for logical data model
  - Step 2.2 Validate relations using normalization
  - Step 2.3 Validate relations against user transactions
  - Step 2.4 Define integrity constraints

# Overview Database Design Methodology (4 to 7)

- Step 2.5 Review logical data model with user
- Step 2.6 Merge logical data models into global model  
(optional) [Assignment](#) [Project](#) [Exam](#) [Help](#)
- Step 2.7 Check for future growth  
<https://powcoder.com>

Add WeChat powcoder

# Overview Database Design Methodology (5 to 7)

Physical database design for relational database

- Step 3 Translate logical data model for target DBMS
  - Step 3.1 Design base relations
  - Step 3.2 Design representation of derived data
  - Step 3.3 Design general constraints

# Overview Database Design Methodology (6 to 7)

- Step 4 Design file organizations and indexes
  - Step 4.1 Analyze transactions
  - Step 4.2 Choose file organization
  - Step 4.3 Choose indexes  
<https://powcoder.com>
  - Step 4.4 Estimate disk space requirements  
[Add WeChat powcoder](#)

# Overview Database Design Methodology (7 to 7)

- Step 5 Design user views
- Step 6 Design security mechanisms
- Step 7 Consider the introduction of controlled redundancy
- Step 8 Monitor and tune the operational system

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 1 Build Conceptual Data (1 to 3)

- To build a conceptual data model of the data requirements of the enterprise.
  - Model entity types, relationship types, attributes and attribute domains, primary and alternate keys, and integrity constraints.
- Step 1.1 Identify entity types
  - To identify the required entity types.
- Step 1.2 Identify relationship types
  - To identify the important relationships that exist between the entity types.

# Step 1 Build Conceptual Data (2 to 3)

- Step 1.3 Identify and associate attributes with entity or relationship types
  - To associate attributes with the appropriate entity or relationship types and document the details of each attribute. [Assignment Project Example Help](https://powcoder.com) <https://powcoder.com>
- Step 1.4 Determine attribute domains
  - To determine domains for the attributes in the data model and document the details of each domain.

# Step 1 Build Conceptual Data (3 to 3)

- Step 1.5 Determine candidate, primary, and alternate key attributes
  - To identify the candidate key(s) for each entity and if there is more than one candidate key, to choose one to be the primary key and the others as alternate keys.
- Step 1.6 Consider use of enhanced modeling concepts (optional step)
  - To consider the use of enhanced modeling concepts, such as specialization / generalization, aggregation, and composition.

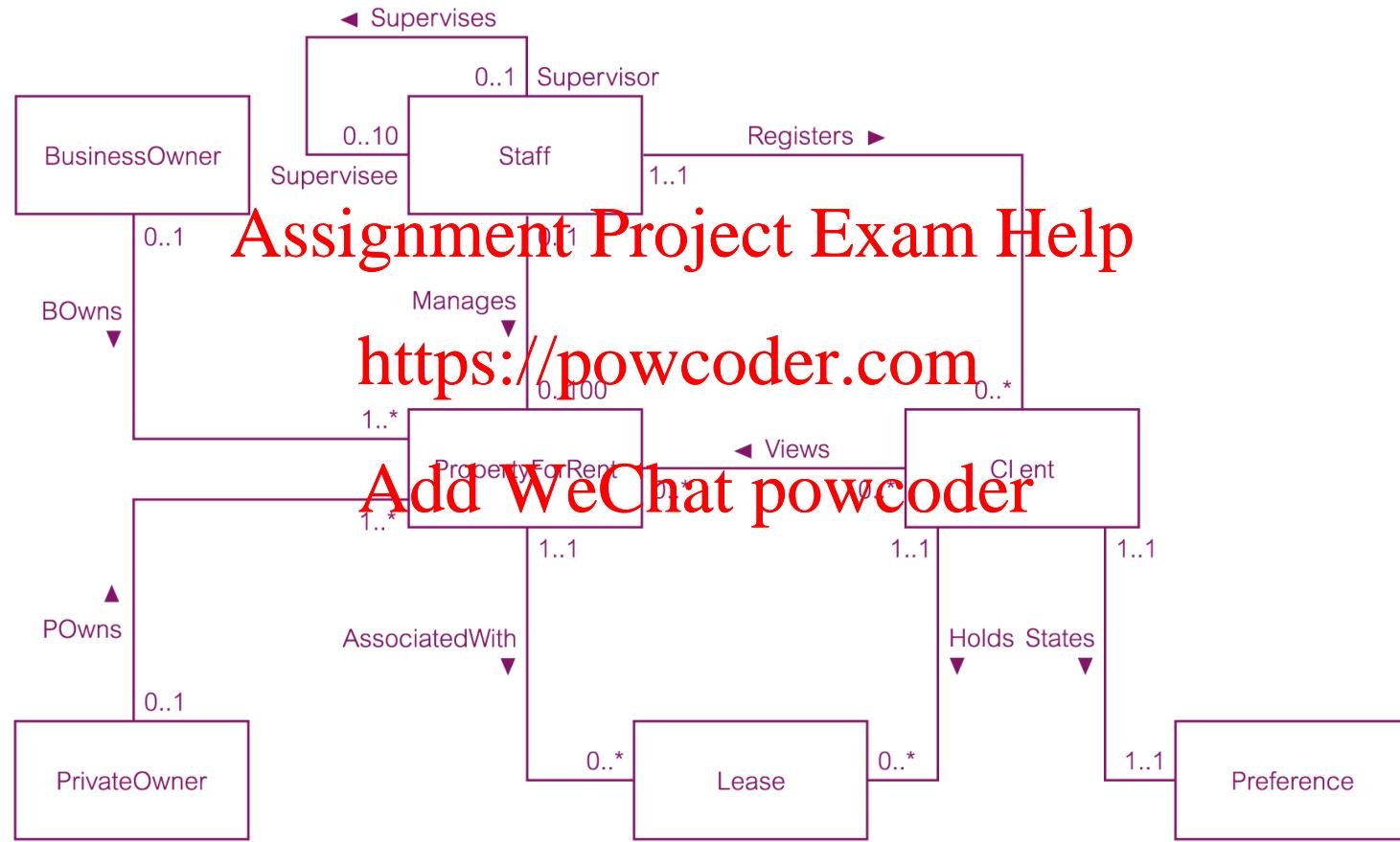
# Step 1 Build Conceptual Data Model

- Step 1.7 Check model for redundancy
  - To check for the presence of any redundancy in the model and ~~Assignment Project Exam Help~~.
- Step 1.8 Validate conceptual model against user transactions
  - To ensure that the conceptual model supports the required transactions.
- Step 1.9 Review conceptual data model with user
  - To review the conceptual data model with the user to ensure that the model is a ‘true’ representation of the data requirements of the enterprise.

# Extract from Data Dictionary for Staff User Views of DreamHome Showing Description of Entities

Entity name	Description	Aliases	Occurrence
<b>Staff</b>	General term describing all staff employed by DreamHome.	Employee	Each member of staff works at one particular branch
<b>PropertyForRent</b>	General term describing all staff property for rent.	Property	Each property has a single owner and id available at one specific branch, where the property is managed by one member of staff. A property is viewed by many clients and rented by a single client, at any one time.

# First-Cut ER Diagram for Staff User Views of DreamHome



# Extract from Data Dictionary for Staff User Views of DreamHome Showing Description of Relationships

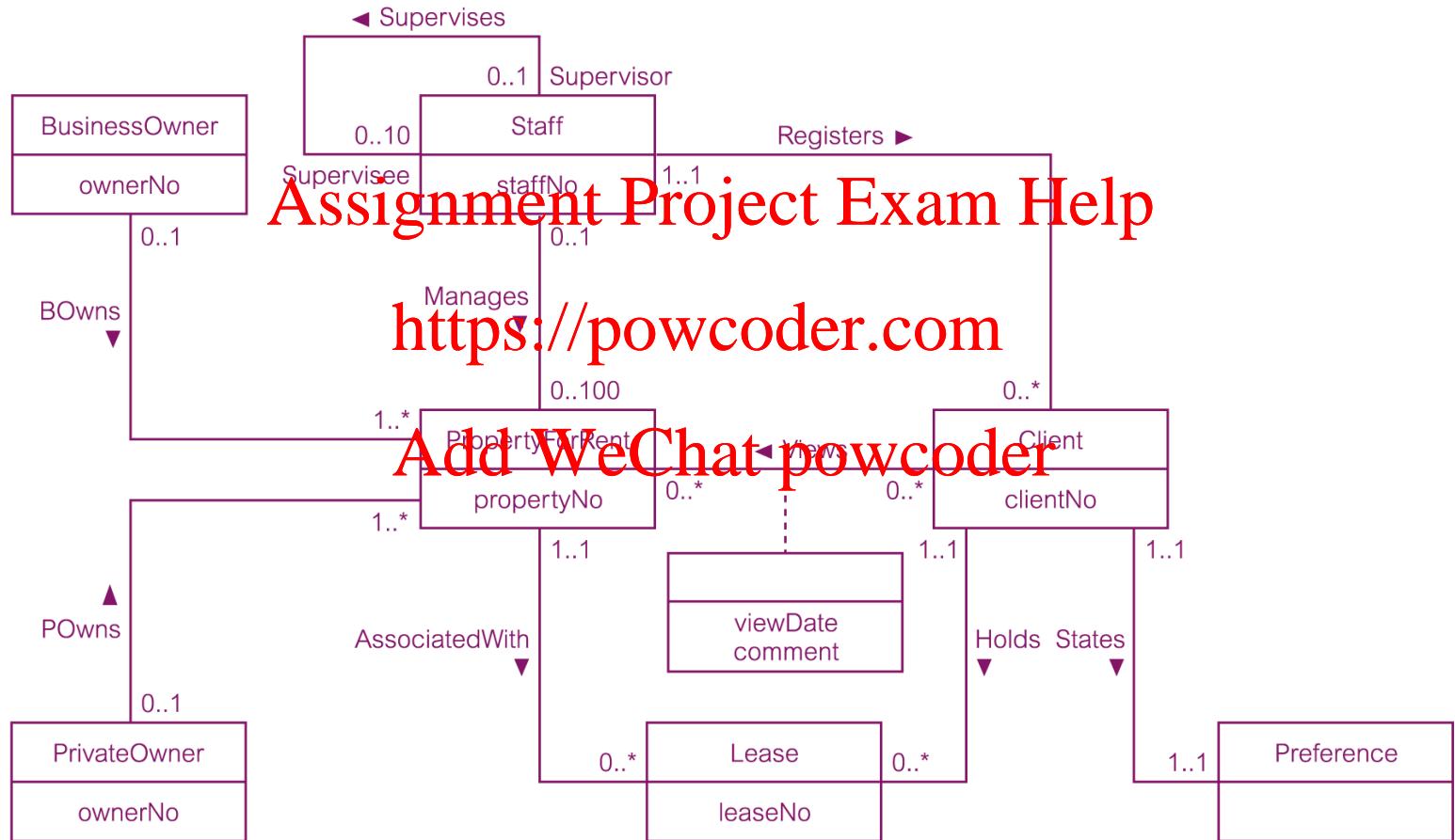
Entity name	Multiplicity	Relationship	Multiplicity	Entity name
Staff	0..1 0..1	Manages Supervises	0..100 0..10	PropertyForRent
PropertyForRent	1..1	AssociatedWith	0..*	Lease

Add WeChat powcoder

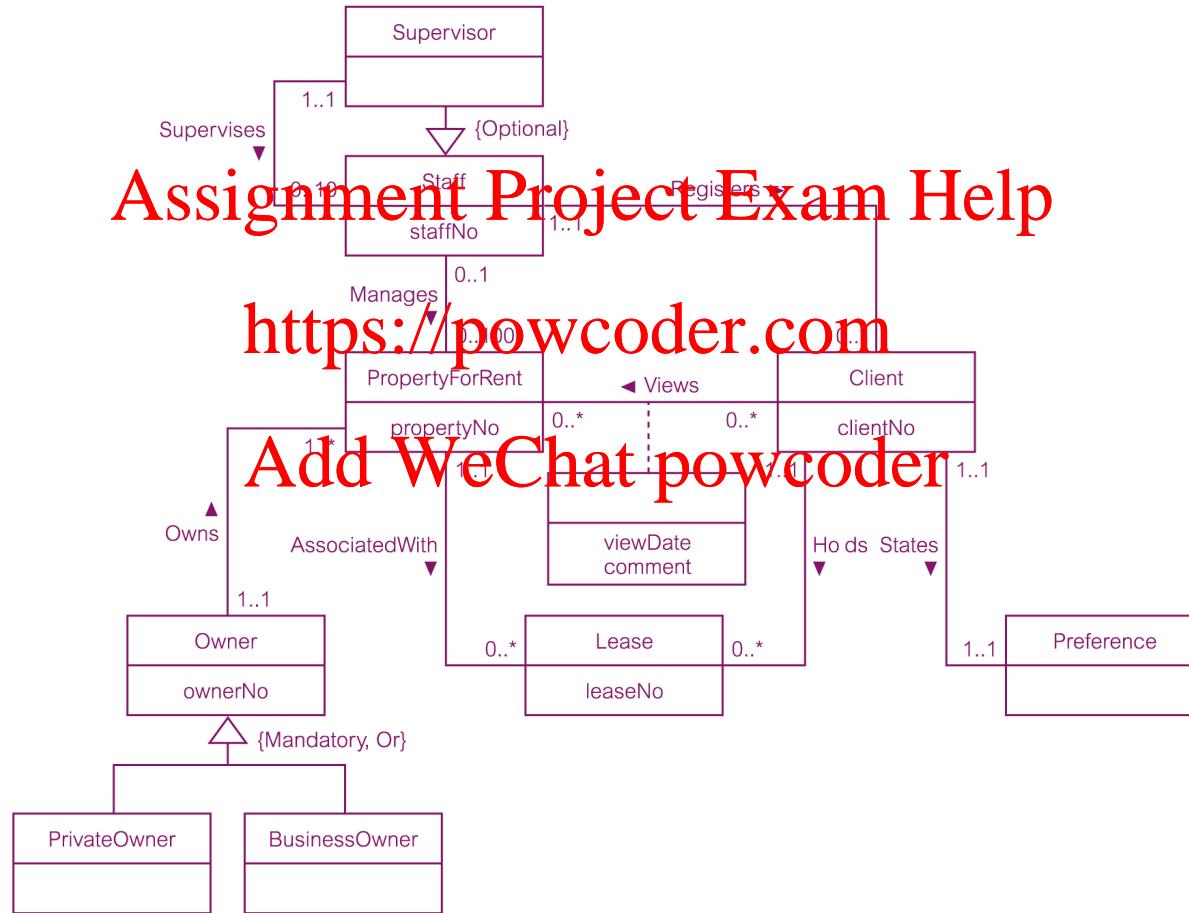
# Extract from Data Dictionary for Staff User Views of DreamHome Showing Description of Attributes

Entity name	Attributes	Description	Data Type & Length	Nulls	Multi-valued	...
Staff	staffNo name fname lname position sex DOB	Uniquely identifies a member of staff First name of staff Last name of staff Job title of member of staff Gender of member of staff Date of birth of member of staff	5 variable characters 15 variable characters 15 variable characters 10 variable characters 1 character (M or F) Date	No No No No Yes Yes	No No No No No No	
PropertyForRent	propertyNo	Uniquely identifies a property to rent	5 variable characters	No	No	

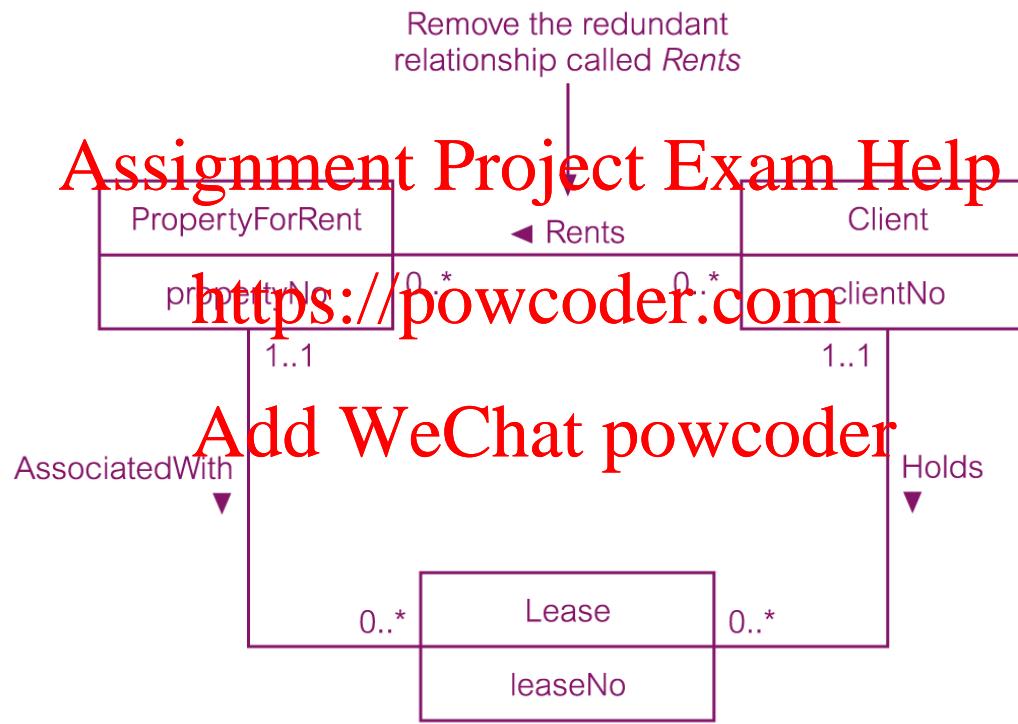
# ER Diagram for Staff User Views of Dream Home with Primary Keys Added



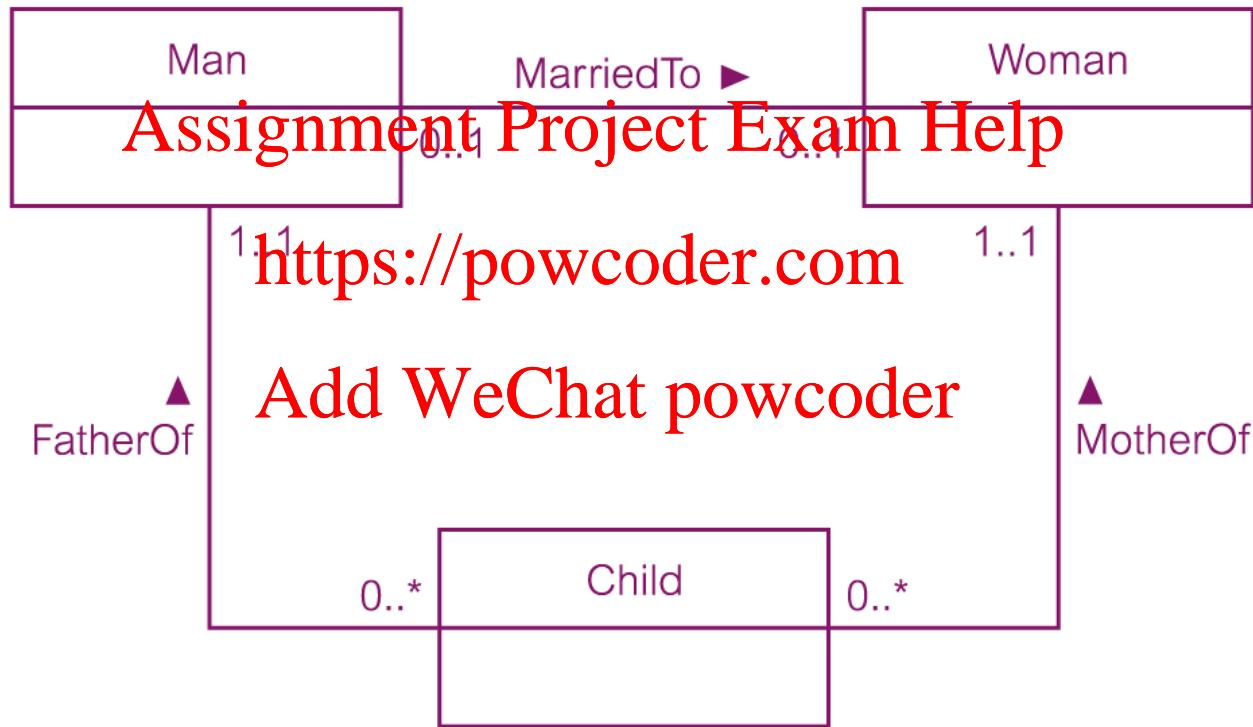
# Revised ER Diagram for Staff User Views of Dream Home with Specialization / Generalization



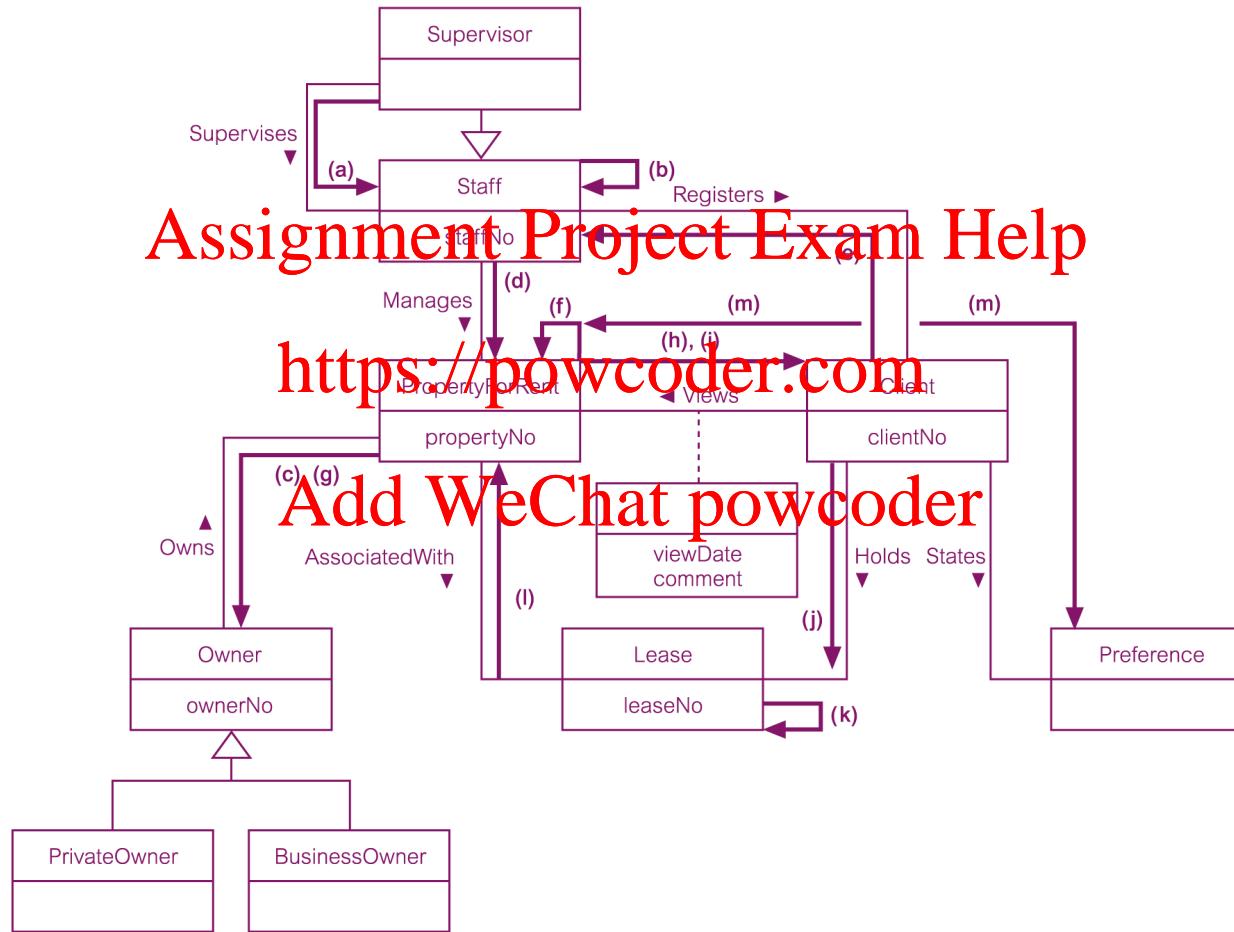
# Example of Removing a Redundant Relationship Called Rents



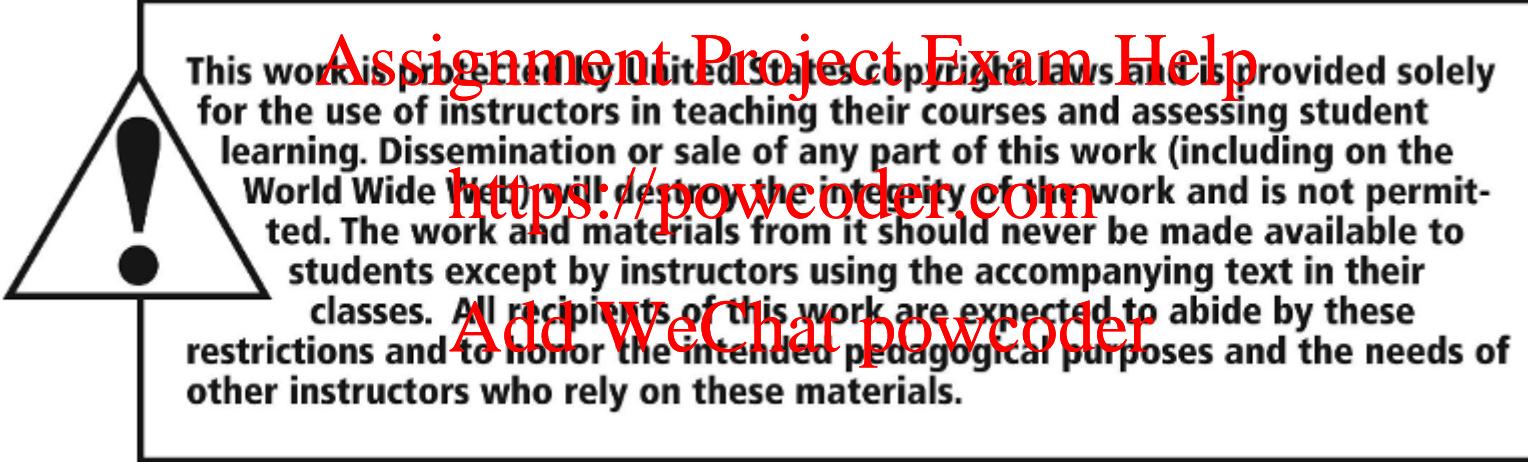
# Example of a Non-Redundant Relationship FatherOf



# Using Pathways to Check That the Conceptual Model Supports the User Transactions

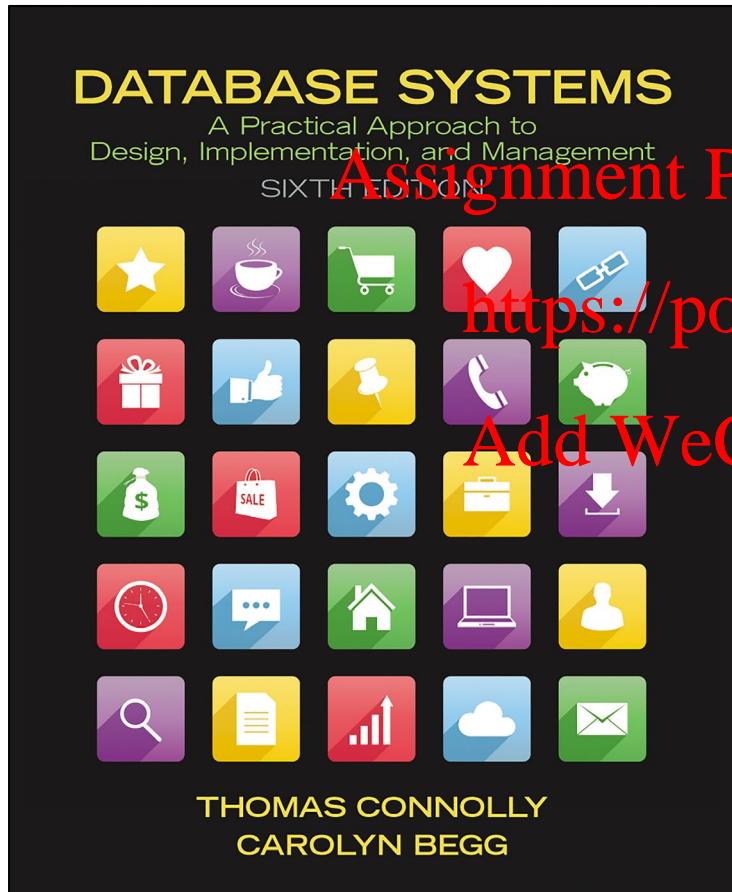


# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 17

<https://powcoder.com>  
Methodology Logical  
Database Design for the  
Relational Model  
Add WeChat powcoder

# Learning Objectives (1 of 2)

**17.1** How to derive a set of relations from a conceptual data model.

**17.2** How to validate these relations using the technique of normalization.

[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

# Learning Objectives (2 of 2)

**17.3** How to validate a logical data model to ensure it supports the required transactions.

**17.4** How to merge local logical data models based on one or more user views into a global logical data model that represents all user views.

**17.5** How to ensure that the final logical data model is a true and accurate representation of the data requirements of the enterprise.

# Step 2 Build and Validate Logical Data Model (1 of 2)

- To translate the conceptual data model into a logical data model and then to validate this model to check that it is structurally correct using normalization and supports the required transactions.

<https://powcoder.com>

Add WeChat powcoder

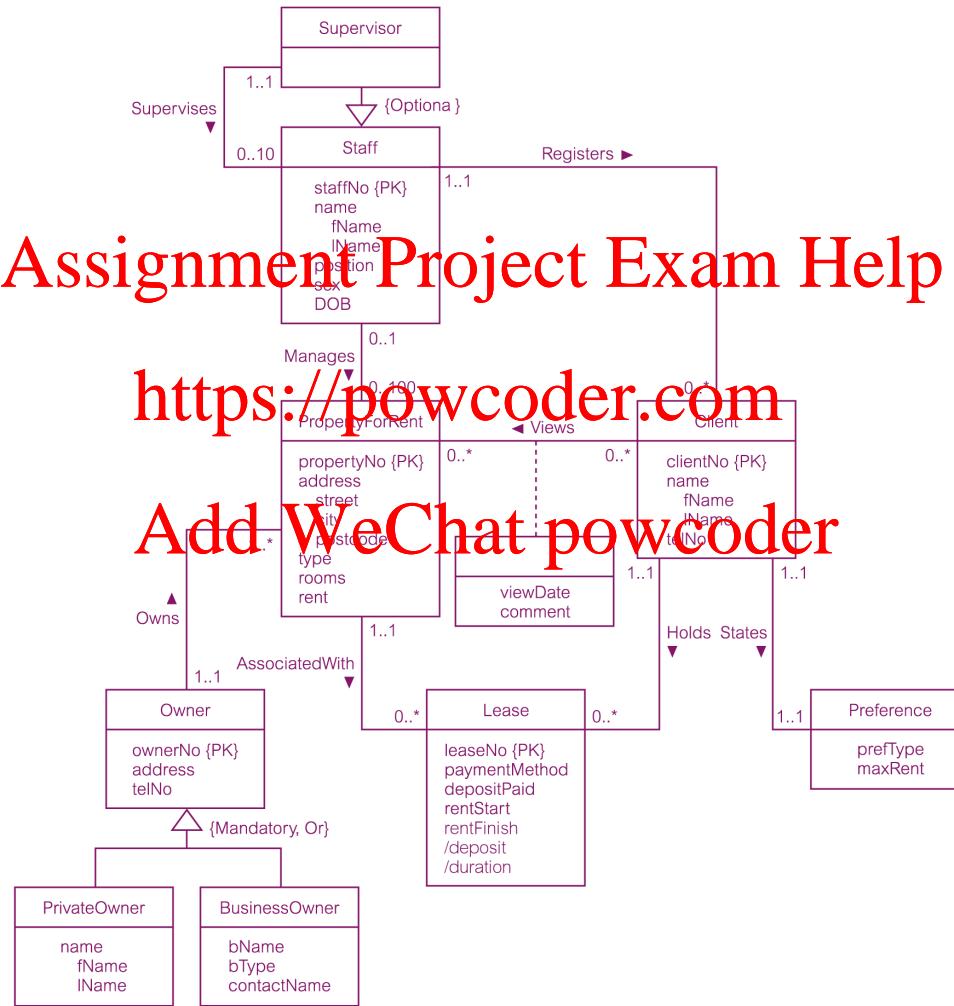
# Step 2 Build and Validate Logical Data Model (2 of 2)

- Step 2.1 Derive relations for logical data model
  - To create relations for the logical data model to represent entities, project relationships, and attributes that have been identified.

<https://powcoder.com>

Add WeChat powcoder

# Conceptual Data Model for Staff View Showing All Attributes



# Step 2.1 Derive Relations for Logical Data Model (1 of 10)

- **(1) Strong entity types**
  - For each strong entity in the data model, create a relation that includes all the simple attributes of that entity. For composite attributes, include only the constituent simple attributes. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com>

## **(2) Weak entity types** [Add WeChat powcoder](#)

- For each weak entity in the data model, create a relation that includes all the simple attributes of that entity. The primary key of a weak entity is partially or fully derived from each owner entity and so the identification of the primary key of a weak entity cannot be made until after all the relationships with the owner entities have been mapped.

# Step 2.1 Derive Relations for Logical Data Model (2 of 10)

- **(3) One-to-many (1: \*) binary relationship types**
  - For each 1: \* binary relationship, the entity on the ‘one side’ of the relationship is designated as the parent entity and the entity on the ‘many side’ is designated as the child entity. To represent this relationship, post a copy of the primary key attribute(s) of parent entity into the relation representing the child entity, to act as a foreign key.

# Step 2.1 Derive Relations for Logical Data Model (3 of 10)

- (4) **One-to-one (1:1) binary relationship types**
  - Creating relations to represent a 1:1 relationship is more complex as the cardinality cannot be used to identify the parent and child entities in a relationship. Instead, the participation constraints are used to provide whether it is best to represent the relationship by combining the entities involved into one relation or by creating two relationships and posting a copy of the primary key from one relation to the other.
  - Consider the following
    - **(a) mandatory** participation on **both** sides of 1:1 relationship;
    - **(b) mandatory** participation on **one** side of 1:1 relationship;
    - **(c) optional** participation on **both** sides of 1:1 relationship;

# Step 2.1 Derive Relations for Logical Data Model (4 of 10)

- **(a) Mandatory participation on both sides of 1:1 relationship**
  - Combine entities involved into one relation and choose one of the primary keys of original entities to be primary key of the new relation, while the other (if one exists) is used as an alternate key.  
<https://powcoder.com>
- **(b) Mandatory participation on one side of a 1:1 relationship**
  - Identify parent and child entities using participation constraints. Entity with optional participation in relationship is designated as parent entity, and entity with mandatory participation is designated as child entity. A copy of primary key of the parent entity is placed in the relation representing the child entity. If the relationship has one or more attributes, these attributes should follow the posting of the primary key to the child relation.

# Step 2.1 Derive Relations for Logical Data Model (5 of 10)

- (c) Optional participation on both sides of a 1:1 relationship
  - In this case, the designation of the parent and child entities is arbitrary unless we can find out more about the relationship that can help a decision to be made one way or the other.

Add WeChat powcoder

# Step 2.1 Derive Relations for Logical Data Model (6 of 10)

- (5) One-to-one (1:1) recursive relationships
  - For a 1:1 recursive relationship, follow the rules for participation as described above for a 1:D relationship.  
**Assignment Project Exam Help**  
<https://powcoder.com>  
**Add WeChat powcoder**
  - mandatory participation on both sides, represent the recursive relationship as a single relation with two copies of the primary key.
  - mandatory participation on only one side, option to create a single relation with two copies of the primary key, or to create a new relation to represent the relationship. The new relation would only have two attributes, both copies of the primary key. As before, the copies of the primary keys act as foreign keys and have to be renamed to indicate the purpose of each in the relation.
  - optional participation on both sides, again create a new relation as described above.

# Step 2.1 Derive Relations for Logical Data Model (7 of 10)

- **(6) Superclass/subclass relationship types**
  - Identify superclass entity as parent entity and subclass entity as the child entity. There are various options on how to represent such a relationship as one or more relations.
  - The selection of the most appropriate option is dependent on a number of factors such as the disjointness and participation constraints on the superclass/subclass relationship, whether the subclasses are involved in distinct relationships, and the number of participants in the superclass/subclass relationship.

# Guidelines for Representation of Superclass/Subclass Relationship

Participation Constraint	Disjoint Constraint	Relations Required
Mandatory	Nondisjoint {And}	Single relation (with one or more discriminators to distinguish the type of each tuple)
Optional	Nondisjoint {And} <a href="https://powcoder.com">https://powcoder.com</a>	Two relations: one relation for superclass and one relation for all subclasses (with one or more discriminators to distinguish the type of each tuple)
Mandatory	Disjoint {Or}	Many relations: one relation for each combined superclass/subclass
Optional	Disjoint {Or}	Many relations: one relation for superclass and one for each subclass

# Representation of Superclass/Subclass Relationship Based on Participation and Disjointness (1 of 3)

## Option 1 – Mandatory, nondisjoint

**AllOwner** (ownerNo, address, telNo, fName, lName, bName, bType,  
contactName, pOwnerFlag, bOwnerFlag)

Assignment Project Exam Help

**Primary Key** ownerNo

<https://powcoder.com>

## Option 2 – Optional, nondisjoint

**Owner** (ownerNo, address, telNo)

Add WeChat powcoder

**Primary Key** ownerNo

**Owner Details** (ownerNo, fName, lName, bName, bType, contactName,  
pOwnerFlag, bOwnerFlag)

**Primary Key** ownerNo

**Foreign Key** ownerNo **references** Owner(ownerNo)

# Representation of Superclass/Subclass Relationship Based on Participation and Disjointness (2 of 3)

## Option 3 – Mandatory, disjoint

**PriavteOwner** (ownerNo, fName, lName, address, telNo)

**Primary Key** ownerNo **Assignment Project Exam Help**

**BusinessOwner** (ownerNo, bName, bType, contactName, address, telNo)  
<https://powcoder.com>

**Primary Key** ownerNo **Add WeChat powcoder**

## Option 4 – Optional, disjoint

**Owner** (ownerNo, address, telNo)

**Primary Key** ownerNo

**PrivateOwner** (ownerNo, fName, lName)

**Primary Key** ownerNo

# Representation of Superclass/Subclass Relationship Based on Participation and Disjointness (3 of 3)

**ForeignKey** ownerNo **references** Owner(ownerNo)

**BusinessOwner** (ownerNo, bName, bType, contactName)

**Primary Key** ownerNo **Assignment Project Exam Help**

**ForeignKey** ownerNo **references** Owner(ownerNo)  
<https://powcoder.com>

Add WeChat powcoder

# Step 2.1 Derive Relations for Logical Data Model (8 of 10)

- **(7) Many-to-many (\* : \*) binary relationship types**
  - Create a relation to represent the relationship and include any attributes that are part of the relationship. We post a copy of the primary key attribute(s) of the entities that participate in the relationship into the new relation, to act as foreign keys. These foreign keys will also form the primary key of the new relation, possibly in combination with some of the attributes of the relationship.

# Step 2.1 Derive Relations for Logical Data Model (9 of 10)

- (8) Complex relationship types
  - Create a relation to represent the relationship and include any attributes that are part of the relationship. Post a copy of the primary key attribute(s) of the entities that participate in the complex relationship into the new relation, to act as foreign keys. Any foreign keys that represent a ‘many’ relationship (for example, 1..\*, 0..\*) generally will also form the primary key of this new relation, possibly in combination with some of the attributes of the relationship.

# Step 2.1 Derive Relations for Logical Data Model (10 of 10)

- **(9) Multi-valued attributes**
  - Create a new relation to represent multi-valued attribute and include primary key of entity in new relation, to act as a foreign key. Unless the multi-valued attribute is itself an alternate key of the entity, the primary key of the new relation is the combination of the multi-valued attribute and the primary key of the entity.

# Summary of How to Map Entities and Relationships to Relations (1 of 2)

Entity/Relationship	Mapping
Strong entity	Create relation that includes all simple attributes.
Weak entity	Create relation that includes all simple attributes (primary key still has to be identified after the relationship with each owner entity has been mapped).
1: * binary relationship	<p><a href="https://powcoder.com">https://powcoder.com</a></p> <p>Add WeChat powcoder</p> <p>Post primary key of entity on the “one” side to act as foreign key in relation representing entity on the “many” side. Any attributes of relationship are also posted to the “many” side.</p>
1:1 binary relationship:	
(a) Mandatory participation on both sides	Combine entities into one relation.
(b) Mandatory participation on one side	Post primary key of entity on the “optional” side to act as foreign key in relation representing entity on the “mandatory” side.
(c) Optional participation on both sides	Arbitrary without further information.
Superclass/subclass relationship	See Table 17.1.(see slide 14)

# Summary of How to Map Entities and Relationships to Relations (2 of 2)

Entity/Relationship	Mapping
* : * binary relationship, complex Relationship	Create a relation to represent the relationship and include any attributes of the relationship. Post a copy of the primary keys from each of the owner entities into the new relation to act as foreign keys.  <a href="https://powcoder.com">Assignment Project Exam Help https://powcoder.com</a>
Multi-valued attribute	Create a relation to represent the multi-valued attribute and post a copy of the primary key of the owner entity into the new relation to act as a foreign key.  <a href="#">Add WeChat powcoder</a>

# Relations for the Staff User Views of DreamHome

<p><b>Staff</b> (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)</p> <p><b>Primary Key</b> staffNo</p> <p><b>Foreign Key</b> supervisorStaffNo <b>references</b> Staff(staffNo)</p>	<p><b>PrivateOwner</b> (ownerNo, fName, lName, address, telNo)</p> <p><b>Primary Key</b> ownerNo</p>
<p><b>BusinessOwner</b> (ownerNo, bName, bType, contactName, address, telNo)</p> <p><b>Primary Key</b> ownerNo</p> <p><b>Alternate Key</b> bName</p> <p><b>Alternate Key</b> telNo</p>	<p><b>Client</b> (clientNo, fName, lName, telNo, prefType, maxRent, staffNo)</p> <p><b>Primary Key</b> clientNo</p> <p><b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p>
<p><b>PropertyForRent</b> (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)</p> <p><b>Primary Key</b> propertyNo</p> <p><b>Foreign Key</b> ownerNo <b>references</b> PrivateOwner(ownerNo) and BusinessOwner(ownerNo)</p> <p><b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p>	<p><b>Viewing</b> (clientNo, propertyNo, dateView, comment)</p> <p><b>Primary Key</b> clientNo, propertyNo</p> <p><b>Foreign Key</b> clientNo <b>references</b> Client(clientNo)</p> <p><b>Foreign Key</b> propertyNo <b>references</b> PropertyForRent(propertyNo)</p>
<p><b>Lease</b> (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)</p> <p><b>Primary Key</b> leaseNo</p> <p><b>Alternate Key</b> propertyNo, rentStart</p> <p><b>Alternate Key</b> clientNo, rentStart</p> <p><b>Foreign Key</b> clientNo <b>references</b> Client(clientNo)</p> <p><b>Foreign Key</b> propertyNo <b>references</b> PropertyForRent(propertyNo)</p> <p><b>Derived</b> deposit (PropertyForRent.rent*2)</p> <p><b>Derived</b> duration (rentFinish – rentStart)</p>	

# Step 2.2 Validate Relations Using Normalization

- To validate the relations in the logical data model using normalization.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 2.3 Validate Relations Against User Transactions

- To ensure that the relations in the logical data model support the required transactions.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## Step 2.4 Check Integrity Constraints

- To check integrity constraints are represented in the logical data model. This includes identifying:
  - Required data [Assignment Project Exam Help](#)
  - Attribute domain constraints <https://powcoder.com>
  - Multiplicity
  - Entity integrity [Add WeChat powcoder](#)
  - Referential integrity
  - General constraints

# Referential Integrity Constraints for Relations in Staff User Views of DreamHome (1 of 2)

**Staff** (staffNo, fName, lName, position, sex, DOB, supervisorStaffNo)

**Primary Key** staffNo

**Foreign Key** supervisorStaffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

**Client** (clientNo, fName, lName, telNo, eMail, prefType, maxRent, staffNo)

**Primary key** clientNo

Assignment Project Exam Help

**Alternate Key** eMail

**Foreign Key** staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE NO ACTION

<https://powcoder.com>

**PropertyForRent** (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo)

**Primary key** propertyNo

**Foreign Key** ownerNo **references** PrivateOwner(ownerNo) and BusinessOwner(ownerNo) ON UPDATE CASCADE  
ON DELETE NO ACTION

Add WeChat powcoder

**Foreign Key** staffNo **references** Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL

**Viewing** (clientNo, propertyNo, dateView, comment)

**Primary Key** clientNo, propertyNo

**Foreign Key** clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION

**Foreign Key** propertyNo **references** PropertyForRent(propertyNo) ON UPDATE CASCADE ON DELETE CASCADE

# Referential Integrity Constraints for Relations in Staff User Views of DreamHome (2 of 2)

**Lease** (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)

**Primary Key** leaseNo

**Alternate Key** propertyNo, rentStart

**Alternate Key** clientNo, rentStart

**Foreign Key** clientNo **references** Client(clientNo) ON UPDATE CASCADE ON DELETE NO ACTION

**Foreign Key** propertyNo **references** PropertyToRent(propertyNo) ON UPDATE CASCADE ON DELETE NO ACTION

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 2.5 Review Logical Data Model with User

- To review the logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of the enterprise

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

## Step 2.6 Merge Logical Data Models into Global Model (Optional Step)

- To merge logical data models into a single global logical data model that represents all user views of a database.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 2.6.1 Merge Local Logical Data Models into Global Model

- To merge local logical data model into a single global logical data model.
- This activities in this step include:
  - Step 2.6.1 Merge local logical data models into global model
  - Step 2.6.2 Validate global logical data model
  - Step 2.6.3 Review global logical data model with users.

# Step 2.6.1 Merge Logical Data Models into a Global Model (1 of 2)

- Tasks typically includes:
  - (1) Review the names and contents of entities/~~Assignment Project Exam Help~~.
  - (2) Review the names and contents of relationships/foreign keys.
  - (3) Merge entities/relations from the local data models
  - (4) Include (without merging) entities/relations unique to each local data model
  - (5) Merge relationships/foreign keys from the local data models.

## Step 2.6.1 Merge Logical Data Models into a Global Model (2 of 2)

- (6) Include (without merging) relationships/foreign keys unique to each local data model.
- (7) Check for missing entities/relations and relationships/foreign keys.  
<https://powcoder.com>
- (8) Check foreign keys.
- (9) Check Integrity Constraints.  
[Add WeChat powcoder](#)
- (10) Draw the global ER/relation diagram
- (11) Update the documentation.

## Step 2.6.2 Validate Global Logical Data Model

- To validate the relations created from the global logical data model using the technique of normalization and to ensure they support the required transactions, if necessary.

<https://powcoder.com>

Add WeChat powcoder

## Step 2.6.3 Review Global Logical Data Model with Users

- To review the global logical data model with the users to ensure that they consider the model to be a true representation of the data requirements of an enterprise.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Relations for the Branch User Views of DreamHome (1 of 2)

<p><b>Branch</b> (branchNo, street, city, postcode, mgrStaffNo) <b>Primary Key</b> branchNo <b>Alternate Key</b> postcode <b>Foreign Key</b> mgrStaffNo <b>references</b> Manager(staffNo)</p>	<p><b>Telephone</b> (telNo, branchNo) <b>Primary Key</b> telNo <b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>
<p><b>Staff</b> (staffNo, name, position, salary, supervisor StaffNo, branchNo) <b>Primary Key</b> staffNo <b>Foreign Key</b> supervisor StaffNo <b>references</b> Staff(staffNo) <b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>	<p><b>Manager</b> (staffNo, mgrStartDate, bonus) <b>Primary Key</b> staffNo <b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p>
<p><b>Private Owner</b> (ownerNo, name, address, telNo) <b>Primary Key</b> ownerNo</p>	<p><b>Business Owner</b> (bName, bType, contactName, address, telNo) <b>Primary Key</b> bName <b>Alternate Key</b> telNo</p>
<p><b>PropertyForRent</b> (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, bName, branchNo) <b>Primary Key</b> propertyNo <b>Foreign Key</b> ownerNo <b>references</b> PrivateOwner(ownerNo) <b>Foreign Key</b> bName <b>references</b> BusinessOwner(bName) <b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo) <b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>	<p><b>Client</b> (clientNo, name, telNo, prefType, maxRent) <b>Primary Key</b> clientNo</p>

# Relations for the Branch User Views of DreamHome (2 of 2)

<p><b>Lease</b> (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo)</p> <p><b>Primary Key</b> leaseNo</p> <p><b>Alternate Key</b> propertyNo, rentStart</p> <p><b>Alternate Key</b> clientNo, rentStart</p> <p><b>ForeignKey</b> clientNo <b>references</b> Client(clientNo)</p> <p><b>ForeignKey</b> propertyNo <b>references</b> PropertyForRent(propertyNo)</p> <p><b>Derived</b> deposit (PropertyForRent.rent*2)</p> <p><b>Derived</b> duration (rentFinish - rentStart)</p>	<p><b>Registration</b> (clientNo, branchNo, staffNo, dateJoined)</p> <p><b>Primary Key</b> clientNo</p> <p><b>Foreign Key</b> clientNo <b>references</b> Client(clientNo)</p> <p><b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p> <p><b>Foreign Key</b>, staffNo <b>references</b> Staff(staffNo)</p>
<p><b>Advert</b> (propertyNo, newspaperName, dateAdvert, cost)</p> <p><b>Primary Key</b> propertyNo, newspaperName, dateAdvert</p> <p><b>Foreign Key</b> propertyNo <b>references</b> PropertyForRent(propertyNo)</p> <p><b>Foreign Key</b> newspaper Name <b>references</b> Newspaper(newspaperName)</p>	<p><b>Newspaper</b> (newspaperName, address, telNo, contactName)</p> <p><b>Primary Key</b> newspaperName</p> <p><b>Alternate Key</b> telNo</p>

# Relations That Represent the Global Logical Data Model for DreamHome (1 of 2)

<p><b>Branch</b> (branchNo, street, city, postcode, mgrStaffNo)</p> <p><b>Primary Key</b> branchNo</p> <p><b>Alternate Key</b> postcode</p> <p><b>Foreign Key</b> mgrStaffNo <b>references</b> Manager(staffNo)</p>	<p><b>Telephone</b> (telNo, branchNo)</p> <p><b>Primary Key</b> telNo</p> <p><b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>
<p><b>Staff</b> (staffNo, fName, Name, position, sex, DOB, salary, supervisorStaffNo, branchNo)</p> <p><b>Primary Key</b> staffNo</p> <p><b>Foreign Key</b> supervisorStaffNo <b>references</b> Staff(staffNo)</p> <p><b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>	<p><b>Manager</b> (staffNo, mgrStartDate, bonus)</p> <p><b>Primary Key</b> staffNo</p> <p><b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p>
<p><b>Private Owner</b> (ownerNo, fName, lName, address, telNo)</p> <p><b>Primary Key</b> ownerNo</p>	<p><b>Business Owner</b> (ownerNo, bName, bType, contactName, address, telNo)</p> <p><b>Primary Key</b> ownerNo</p> <p><b>Alternate Key</b> bName</p> <p><b>Alternate Key</b> telNo</p>
<p><b>PropertyForRent</b> (propertyNo, street, city, postcode, type, rooms, rent, ownerNo, staffNo, branchNo)</p> <p><b>Primary Key</b> propertyNo</p> <p><b>Foreign Key</b> ownerNo <b>references</b> PrivateOwner(ownerNo) and BusinessOwner(ownerNo)</p> <p><b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p> <p><b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo)</p>	<p><b>Viewing</b> (clientNo, propertyNo, dateView, comment)</p> <p><b>Primary Key</b> clientNo, propertyNo</p> <p><b>Foreign Key</b> clientNo <b>references</b> Client(clientNo)</p> <p><b>Foreign Key</b> propertyNo <b>references</b> PropertyForRent(propertyNo)</p>

# Relations That Represent the Global Logical Data Model for DreamHome (2 of 2)

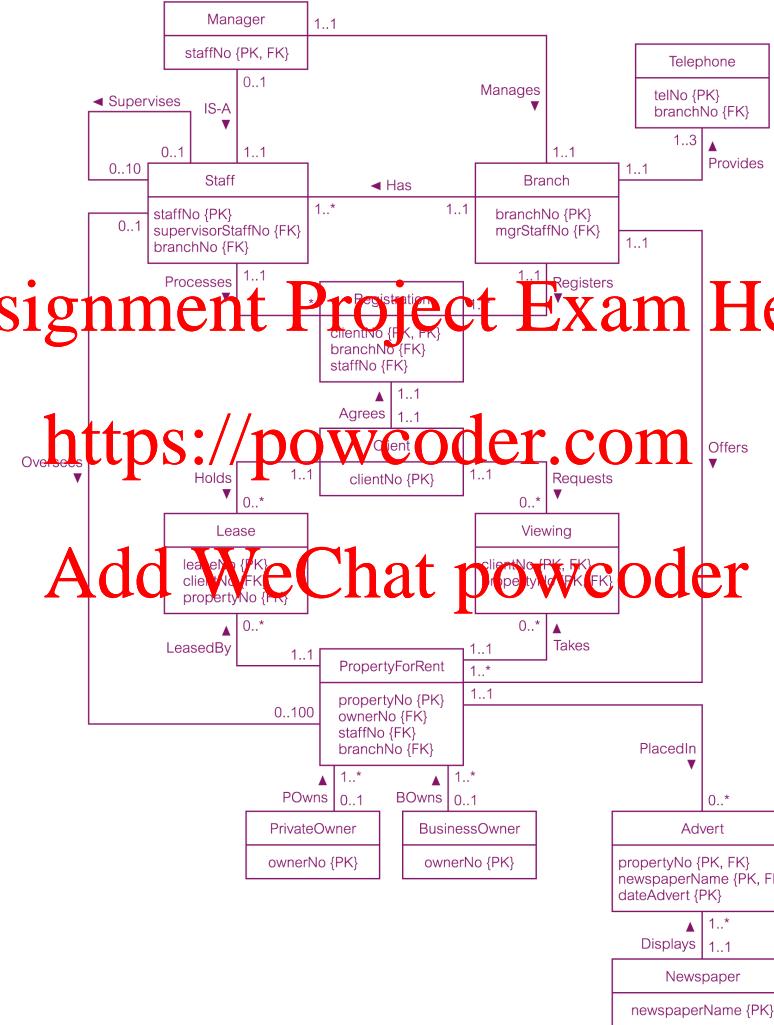
<p><b>Client</b> (clientNo, fName, lName, telNo, eMail, prefType, maxRent) <b>Primary Key</b> clientNo</p>	<p><b>Registration</b> (clientNo, branchNo, staffNo, dateJoined) <b>Primary Key</b> clientNo <b>Foreign Key</b> clientNo <b>references</b> Client(clientNo) <b>Foreign Key</b> branchNo <b>references</b> Branch(branchNo) <b>Foreign Key</b> staffNo <b>references</b> Staff(staffNo)</p>
<p><b>Lease</b> (leaseNo, paymentMethod, depositPaid, rentStart, rentFinish, clientNo, propertyNo) <b>Primary Key</b> leaseNo <b>Alternate Key</b> propertyNo, rentStart <b>Alternate Key</b> clientNo, rentStart <b>Foreign Key</b> clientNo <b>references</b> Client(clientNo) <b>Foreign Key</b> propertyNo <b>references</b> PropertyForRent(propertyNo) <b>Derived</b> deposit (PropertyForRent.rent*2) <b>Derived</b> duration (rentFinish – rentStart)</p>	<p><b>Newspaper</b> (newspaperName, address, telNo, contactName) <b>Primary Key</b> newspaper Name <b>Alternate Key</b> telNo</p>
<p><b>Advert</b> (propertyNo, newspaperName, dateAdvert, cost) <b>Primary Key</b> propertyNo, newspaperName, dateAdvert <b>Foreign Key</b> propertyNo Name <b>references</b> PropertyForRent Newspaper(property No) <b>Foreign Key</b> newspaperName <b>references</b> PropertyForRent Newspaper(property No)</p>	

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Global Relation Diagram for DreamHome

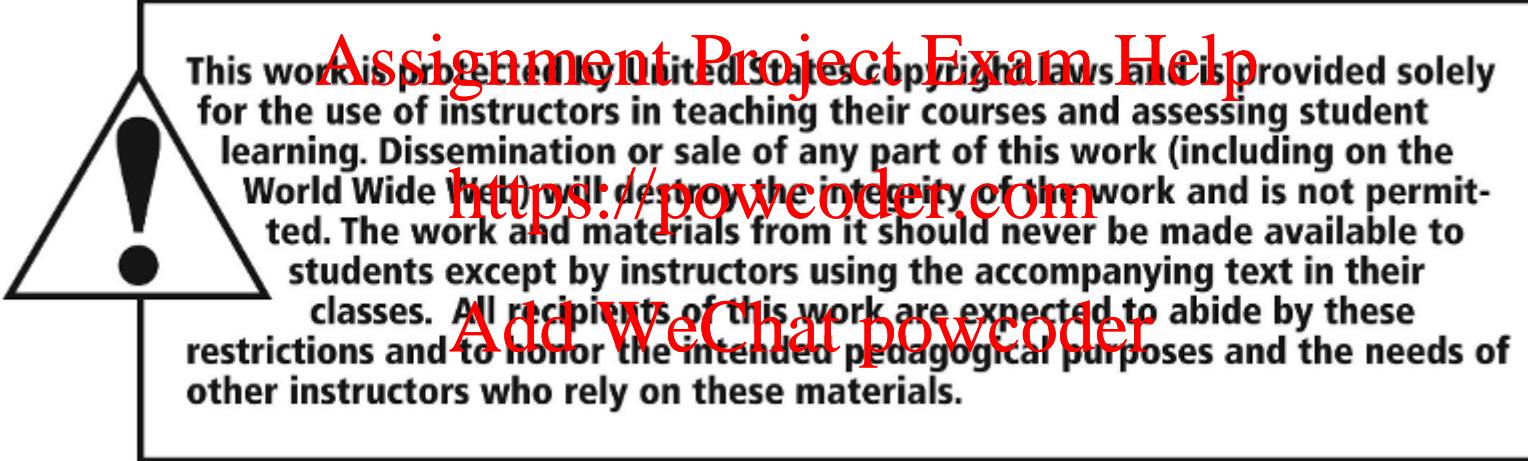


Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 18

<https://powcoder.com>  
Methodology – Physical  
Database Design for  
Relational Databases

Add WeChat powcoder

# Learning Objectives (1 of 2)

- 18.1** Purpose of physical database design.
- 18.2** How to map the logical database design to a physical database design.  
*Assignment Project Exam Help*
- 18.3** How to design base relations for target DBMS.
- 18.4** How to design general constraints for target DBMS.  
*Add WeChat powcoder*

# Learning Objectives (2 of 2)

**18.5** How to select appropriate file organizations based on analysis of transactions.

**18.6** When to use secondary indexes to improve performance. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com>

**18.7** How to estimate the size of the database.

[Add WeChat powcoder](#)

**18.8** How to design user views.

**18.9** How to design security mechanisms to satisfy user requirements.

# Logical v. Physical Database Design

- Sources of information for physical design process includes logical data model and documentation that describes model
- Logical database design is concerned with the **what**, physical database design is concerned with the **how**.

[Assignment Project Exam Help](https://powcoder.com)

[Add WeChat powcoder](https://powcoder.com)

# Physical Database Design

- Process of producing a description of the implementation of the database on secondary storage.
- It describes the base relations, file organizations, and indexes used to achieve efficient access to the data, and any associated integrity constraints and security measures.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Overview of Physical Database Design Methodology (1 of 3)

- Step 3 Translate logical data model for target DBMS
  - Step 3.1 Design base relations
  - Step 3.2 Design representation of derived data
  - Step 3.3 Design general constraints

Add WeChat powcoder  
<https://powcoder.com>

# Overview of Physical Database Design Methodology (2 of 3)

- Step 4 Design file organizations and indexes
  - Step 4.1 Analyze transactions
  - Step 4.2 Choose file organizations
  - Step 4.3 Choose indexes  
<https://powcoder.com>
  - Step 4.4 Estimate disk space requirements  
[Add WeChat powcoder](#)

# Overview of Physical Database Design Methodology (3 of 3)

- Step 5 Design user views
- Step 6 Design security mechanisms
- Step 7 Consider the introduction of controlled redundancy
- Step 8 Monitor and tune operational system

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 3 Translate Logical Data Model for Target DBMS

To produce a relational database schema from the logical data model that can be implemented in the target DBMS.

- Need to know functionality of target DBMS such as how to create base relations and whether the system supports the definition of:
  - PKs, FKs, and AKs;
  - required data – i.e. whether system supports NOT NULL;
  - domains;
  - relational integrity constraints;
  - general constraints.

## Step 3.1 Design Base Relations (1 of 2)

To decide how to represent base relations identified in logical model in target DBMS.

- For each relation, need to define:
  - the name of the relation: <https://powcoder.com>
  - a list of simple attributes in brackets;
  - the PK and, where appropriate, AKs and FKS.
  - referential integrity constraints for any FKS identified.

## Step 3.1 Design Base Relations (2 of 2)

- From data dictionary, we have for each attribute:
  - its domain, consisting of a data type, length, and any constraints on the domain;
  - an optional default value for the attribute;
  - whether it can hold nulls;
  - whether it is derived, and if so, how it should be computed.

# DBDL for the PropertyForRent Relation

Domain PropertyNumber:	variable length character string, length 5	
Domain Street:	variable length character string, length 25	
Domain City:	variable length character string, length 15	
Domain Postcode:	variable length character string, length 8	
Domain PropertyType:	single character, must be one of 'B', 'C', 'D', 'E', 'F', 'H', 'M', 'S'	
Domain PropertyRooms:	integer, in the range 1-15	
Domain PropertyRent:	monetary value in the range 0.00 - 999.99	
Domain OwnerNumber:	variable length character string, length 5	
Domain StaffNumber:	variable length character string, length 5	
Domain BranchNumber:	fixed length character string, length 4	
PropertyForRent(	<a href="https://powcoder.com">https://powcoder.com</a>	
propertyNo	PropertyNumber	NOT NULL,
street	Street	NOT NULL,
city	City	NOT NULL,
postcode	Postcode	NOT NULL,
type	PropertyType	NOT NULL DEFAULT 'F',
rooms	PropertyRooms	NOT NULL DEFAULT 4,
rent	PropertyRent	NOT NULL DEFAULT 600,
ownerNo	OwnerNumber	NOT NULL,
staffNo	StaffNumber,	
branchNo	BranchNumber	NOT NULL,
PRIMARY KEY (propertyNo),		
FOREIGN KEY (staffNo) REFERENCES Staff(staffNo) ON UPDATE CASCADE ON DELETE SET NULL,		
FOREIGN KEY (ownerNo) REFERENCES PrivateOwner(ownerNo) and BusinessOwner(ownerNo)		
ON UPDATE CASCADE ON DELETE NO ACTION,		
FOREIGN KEY (branchNo) REFERENCES Branch(branchNo)		
ON UPDATE CASCADE ON DELETE NO ACTION);		



## Step 3.2 Design Representation of Derived Data (1 of 2)

To decide how to represent any derived data present in logical data model in target DBMS.

- Examine logical data model and data dictionary, and produce list of all derived attributes.  
<https://powcoder.com>
- Derived attribute can be stored in database or calculated every time it is needed.  
[Add WeChat powcoder](#)

## Step 3.2 Design Representation of Derived Data (2 of 2)

- Option selected is based on:
  - additional cost to store the derived data and keep it consistent with the original data from which it is derived;
  - cost to calculate it each time it is required.
- Less expensive option is chosen subject to performance constraints.

# PropertyForRent Relation and Staff Relation with Derived Attribute noOfProperties (1 of 2)

## PropertyForRent

propertyNo	street	city	postcode	type	Rooms	rent	ownerNo	staffNo	branchNo
PA14	16 Holhead	Aberdeen	AB7 5SU	House	6	650	CO46	SA9	B007
PL94	6 Argyll St	London	NW2	Flat	4	400	CO87	SL41	B005
PG4	6 Lawrence St	Glasgow	G11 9QX	Flat	3	350	CO40	SL41	B003
PG36	2 Manor Rd	Glasgow	G32 4QX	Flat	3	375	CO93	SG37	B003
PG21	18 Dale Rd	Glasgow	G12	House	5	600	CO87	SG37	B003
PG16	5 Novar Dr	Glasgow	G12 9AX	Flat	4	450	CO93	SG14	B003

# PropertyForRent Relation and Staff Relation with Derived Attribute noOfProperties (2 of 2)

## Staff

staffNo	fName	lName	branchNo	noOfProperties
SL21	John	White	B005	0
SG37	Ann	Beech	B003	2
SG14	David	Ford	B003	1
SA9	Mary	Howe	B007	1
SG5	Susan	Brand	B003	0
SL41	Julie	Lee	B005	1

## Step 3.3 Design General Constraints

To design the general constraints for target DBMS.

- Some DBMS provide more facilities than others for defining enterprise constraints. Example:

CONSTRAINT StaffNotHandlingTooMuch

```
CHECK (NOT EXISTS (SELECT staffNo  
                   FROM PropertyForRent  
                   GROUP BY staffNo  
                   HAVING COUNT(*) > 100))
```

# Step 4 Design File Organizations and Indexes

To determine optimal file organizations to store the base relations and the indexes that are required to achieve acceptable performance; that is, the way in which relations and tuples will be held on secondary storage.

<https://powcoder.com>

- Must understand the typical **workload** that database must support. [Add WeChat powcoder](https://powcoder.com)

## Step 4.1 Analyze Transactions (1 of 4)

To understand the functionality of the transactions that will run on the database and to analyze the important transactions. **Assignment Project Exam Help**

- Attempt to identify performance criteria, such as:  
<https://powcoder.com>
  - transactions that run frequently and will have a significant impact on performance;
  - transactions that are critical to the business;
  - times during the day/week when there will be a high demand made on the database (called the **peak load**).

## Step 4.1 Analyze Transactions (2 of 4)

- Use this information to identify the parts of the database that may cause performance problems.
- Also need to know high-level functionality of the transactions, such as:  
<https://powcoder.com>
  - attributes that are updated;
  - search criteria used in a query.

## Step 4.1 Analyze Transactions (3 of 4)

- Often not possible to analyze all transactions, so investigate most ‘important’ ones.
- To help identify these can use:
  - **transaction/relation cross-reference matrix**,  
<https://powcoder.com>  
showing relations that each transaction accesses,  
and/or [Add WeChat powcoder](#)
  - **transaction usage map**, indicating which relations  
are potentially heavily used.

## Step 4.1 Analyze Transactions (4 of 4)

- To focus on areas that may be problematic:
  1. Map all transaction paths to relations.
  2. Determine which relations are most frequently accessed by transactions.  
<https://powcoder.com>
  3. Analyze the data usage of selected transactions that involve these relations.  
[Add WeChat powcoder](#)

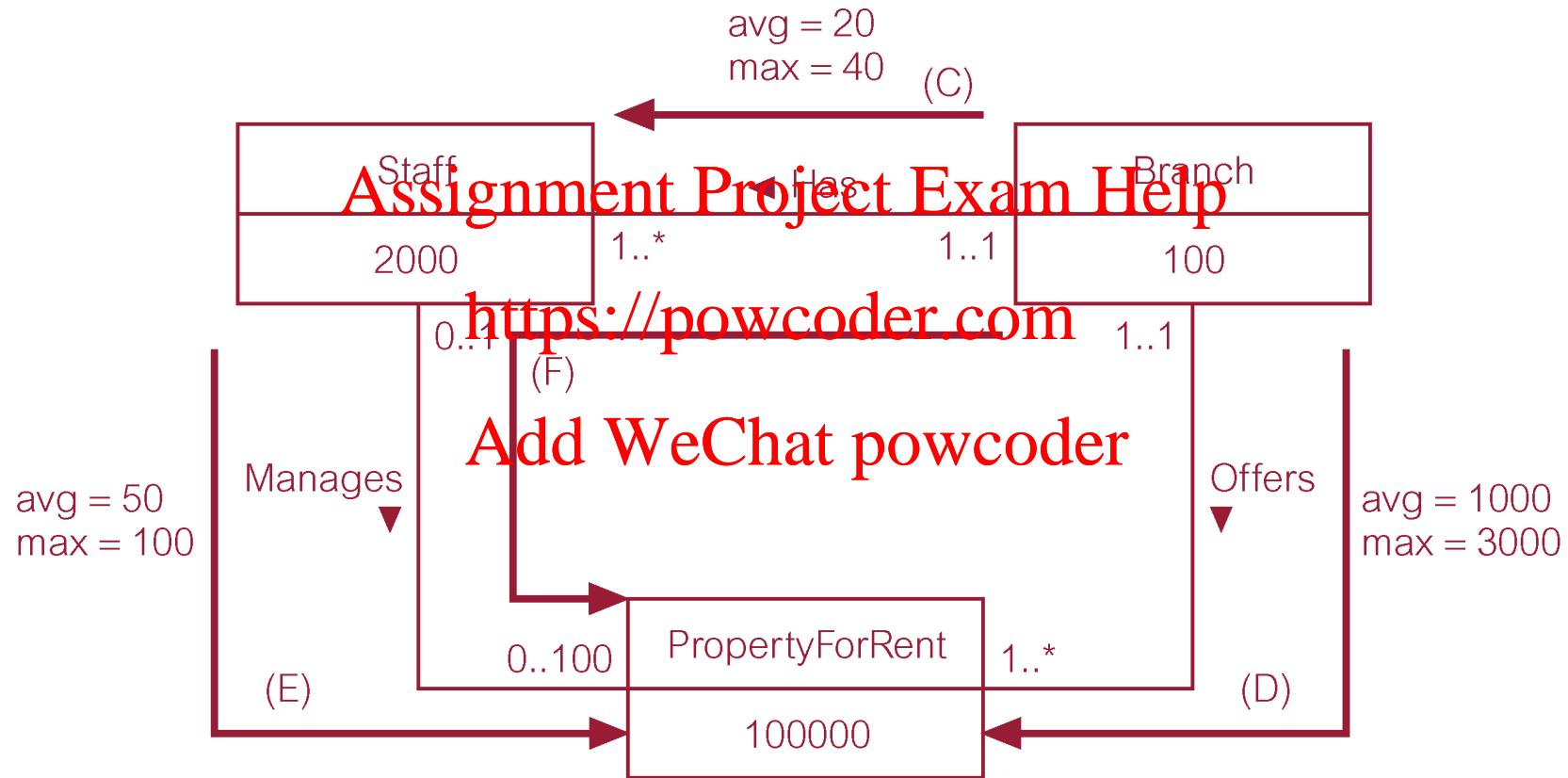
# Cross-Referencing Transactions and Relations

Table 17.1 Cross-referencing transactions and relations.

Transaction/ Relation	(A)	(B)	(C)	(D)	(E)	(F)
	I R U D	R U D	R U D C	R U D C I	R U D	I R U D
Branch			X	X		X
Telephone						
Staff	X		X	X		X
Manager						
PrivateOwner	X					
BusinessOwner	X					
PropertyForRent	X		X X X		X	X
Viewing					X	
Client					X	
Registration						
Lease						
Newspaper						
Advert						

I = Insert; R = Read; U = Update; D = Delete

# Example Transaction Usage Map



# Figure 17.4 Example Transaction Analysis Form

Transaction Analysis Form			1-Sept-2004		
<b>Transaction</b>	(D) List the property number, address, type, and rent of all properties in Glasgow, ordered by rent				
<b>Transaction volume</b>					
Average:	50 per hour				
Peak:	100 per hour (between 17.00 and 19.00 Monday–Saturday)				
<pre>SELECT p.propertyNo, p.address, p.type, p.rent   FROM Branch b   JOIN PropertyForRent p ON     b.branchNo = p.branchNo  WHERE p.city = 'Glasgow'  ORDER BY rent;</pre>					
Predicates: <code>p.city = 'Glasgow'</code> Join attributes: <code>b.branchNo = p.branchNo</code> Ordering attribute: <code>rent</code> Grouping attribute: <code>none</code> Built-in functions: <code>none</code> Attributes updated: <code>none</code>					
<a href="https://powcoder.com">https://powcoder.com</a>					
<p>Assume 4 Glasgow offices</p>					
<b>Access</b>	<b>Entity</b>	<b>Type of Access</b>	<b>No. of References</b>		
			<b>Per Transaction</b>	<b>Avg Per Hour</b>	<b>Peak Per Hour</b>
1	Branch (entry)	R	100	5000	10000
2	PropertyForRent	R	4000–12000	200000–600000	400000–1200000
<b>Total References</b>			<b>4100–12100</b>	<b>205000–605000</b>	<b>410000–1210000</b>

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat [powcoder](https://powcoder.com)

## Step 4.2 Choose File Organizations

To determine an efficient file organization for each base relation.

- File organizations include Heap, Hash, Indexed Sequential Access Method (ISAM), B+-Tree, and Clusters.
- Some DBMSs may not allow selection of file organizations.

## Step 4.3 Choose Indexes (1 of 4)

To determine whether adding indexes will improve the performance of the system.

- One approach is to keep tuples unordered and create as many **secondary indexes** as necessary.  
<https://powcoder.com>

Add WeChat powcoder

## Step 4.3 Choose Indexes (2 of 4)

- Another approach is to order tuples in the relation by specifying a **primary** or **clustering index**.
- In this case, choose the attribute for ordering or clustering the tuples as: [Assignment Project Exam Help](https://powcoder.com)
  - attribute that is used most often for join operations - this makes join operation more efficient, or
  - attribute that is used most often to access the tuples in a relation in order of that attribute.

## Step 4.3 Choose Indexes (3 of 4)

- If ordering attribute chosen is key of relation, index will be a **primary index**; otherwise, index will be a **clustering index**. [Assignment](#) [Project](#) [Exam](#) [Help](#)
- Each relation can only have either a primary index or a clustering index. <https://powcoder.com>
- Secondary indexes provide a mechanism for specifying an additional key for a base relation that can be used to retrieve data more efficiently. [Add WeChat](#) [powcoder](#)

## Step 4.3 Choose Indexes (4 of 4)

- Have to balance overhead involved in maintenance and use of secondary indexes against performance improvement gained when retrieving data.
- This includes:
  - adding an index record to every secondary index whenever tuple is inserted;
  - updating secondary index when corresponding tuple updated;
  - increase in disk space needed to store secondary index;
  - possible performance degradation during query optimization to consider all secondary indexes.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

## Step 4.3 Choose Indexes – Guidelines for Choosing ‘Wish-List’ (1 of 2)

1. Do not index small relations.
2. Index PK of a relation if it is not a key of the file organization.  
*Assignment Project Exam Help*
3. Add secondary index to a FK if it is frequently accessed.  
*https://powcoder.com*
4. Add secondary index to any attribute heavily used as a secondary key.  
*Add WeChat powcoder*
5. Add secondary index on attributes involved in: selection or join criteria; ORDER BY; GROUP BY; and other operations involving sorting (such as UNION or DISTINCT).

## Step 4.3 Choose Indexes – Guidelines for Choosing ‘Wish-List’ (2 of 2)

6. Add secondary index on attributes involved in built-in functions.
7. Add secondary index on attributes that could result in an index-only plan.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
8. Avoid indexing an attribute or relation that is frequently updated.  
[Add WeChat powcoder](#)
9. Avoid indexing an attribute if the query will retrieve a significant proportion of the relation.
10. Avoid indexing attributes that consist of long character strings.

## Step 4.4 Estimate Disk Space Requirements

To estimate the amount of disk space that will be required by the database.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Step 5 Design User Views

To design the user views that were identified during the Requirements Collection and Analysis stage of the database system development life cycle.

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

# Step 6 Design Security Measures

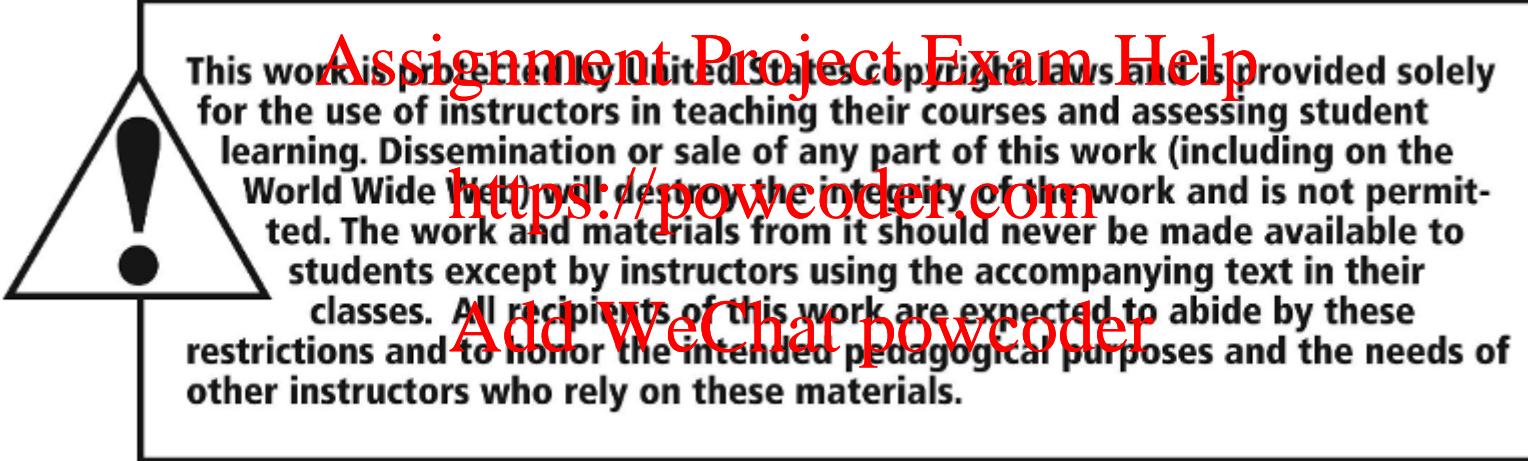
To design the security measures for the database as specified by the users.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Copyright



# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Chapter 9

<https://powcoder.com> Object-Relational DBMSs

Add WeChat powcoder

# Learning Objectives (1 of 2)

**9.1** Requirements for advanced database applications.

**9.2** Why RDBMSs currently are not well suited to supporting such applications.  
*Assignment Project Exam Help*

**9.3** Problems associated with storing objects in a RDB.  
*<https://powcoder.com>*

**9.4** Object-oriented features in SQL:2011:  
*Add WeChat powcoder*

- row types;
- user-defined types and user-defined routines;
- polymorphism;
- inheritance;

# Learning Objectives (2 of 2)

- reference types and object identity;
- collection types (ARRAYs, MULTISets, SETs, and LISTS); **Assignment Project Exam Help**
- extensions to the SQL language to make it computationally complete; **<https://powcoder.com>**
- triggers; **Add WeChat powcoder**
- BLOBs and CLOBs;
- recursion.

## 9.5 Object-oriented extensions to Oracle.

# Advanced Database Applications (1 of 2)

- Computer-Aided Design/Manufacturing (CAD/CAM)
- Computer-Aided Software Engineering (CASE)  
**Assignment Project Exam Help**
- Network Management Systems  
**https://powcoder.com**
- Office Information Systems (OIS) and Multimedia Systems  
**Add WeChat powcoder**
- Digital Publishing
- Geographic Information Systems (GIS)
- Interactive and Dynamic Web sites

# Computer-Aided Design (CAD)

- Stores data relating to mechanical and electrical design, for example, buildings, airplanes, and integrated circuit chips.
- Designs of this type have some common characteristics:
  - Data has many types, each with a small number of instances. <https://powcoder.com>
  - Designs may be very large.
  - Design is not static but evolves through time.
  - Updates are far-reaching.
  - Involves version control and configuration management.
  - Cooperative engineering.

# Advanced Database Applications (2 of 2)

- Computer-Aided Manufacturing (CAM)
  - Stores similar data to CAD, plus data about discrete production
- Computer-Aided Software Engineering (CASE)
  - Stores data about stages of software development lifecycle.

# Network Management Systems

- Coordinate delivery of communication services across a computer network.
- Perform such tasks as network path management, problem management, and network planning.  
<https://powcoder.com>
- Systems handle complex data and require real-time performance and continuous operation.  
[Add WeChat powcoder](#)
- To route connections, diagnose problems, and balance loadings, systems have to be able to move through this complex graph in real-time.

# Office Information Systems (OIS) and Multimedia Systems

- Stores data relating to computer control of information in a business, including electronic mail, documents, invoices, and so on. [Assignment Project Exam Help](https://powcoder.com)
- Modern systems now handle free-form text, photographs, diagrams, audio and video sequences.
- Documents may have specific structure, perhaps described using mark-up language such as SGML, HTML, or XML.

# Digital Publishing

- Can store books, journals, papers, and articles electronically and deliver them over high-speed networks to consumers.
- Can also handle multimedia documents consisting of text, audio, image, and video data and animation.
- Amount of information available to be put online is in the order of petabytes ( $10^{15}$  bytes), making them largest databases DBMS has ever had to manage.

# Geographic Information Systems (GIS)

- GIS database stores spatial and temporal information, such as that used in land management and underwater exploration
- Much of data is derived from survey and satellite photographs, and tends to be very large.
- Searches may involve identifying features based, for example, on shape, color, or texture, using advanced pattern-recognition techniques.

# Interactive and Dynamic Web Sites

- Online catalog for selling clothes maintains preferences for previous visitors and allows visitor to:
  - obtain 3D rendering of any item based on color, size, fabric, etc.;
  - modify rendering to account for movement, illumination, backdrop, occasion, etc.;
  - select accessories to go with the outfit, from items presented in a sidebar;
- Need to handle multimedia content and to interactively modify display based on user preferences and user selections. Added complexity of providing 3D rendering.

# Weaknesses of RDBMSs (1 of 5)

- Poor Representation of “Real World” Entities
  - Normalization leads to relations that do not correspond to entities in “real world”
- Semantic Overloading
  - Relational model has only one construct for representing data and data relationships: the relation.
  - Relational model is **semantically overloaded**.

## Weaknesses of RDBMSs (2 of 5)

- Poor Support for Integrity and General Constraints
- Homogeneous Data Structure
  - Relational model assumes both horizontal and vertical homogeneity.
  - Many RDBMSs now allow **Binary Large Objects (BLOBs)**.

# Weaknesses of RDBMSs (3 of 5)

- Limited Operations
  - RDBMSs only have a fixed set of operations which cannot be extended.
- Difficulty Handling Recursive Queries
  - Extremely difficult to produce recursive queries.
  - Extension proposed to relational algebra to handle this type of query is unary transitive (recursive) closure operation.

# Example - Recursive Query

staffNo	managerstaffNo
S005	S004
S004	S003
S003	S002
S002	S001
S001	NULL

(a)

Assignment Project Exam Help  
<https://powcoder.com>

staffNo	managerstaffNo
S005	S004
S004	S003
S003	S002
S002	S001
S001	NULL
S005	S003
S005	S002
S005	S001
S004	S002
S004	S001
S003	S001

(b)

# Weaknesses of RDBMSs (4 of 5)

- Impedance Mismatch
  - Most DMLs lack **computational completeness**.
  - To overcome this, SQL can be embedded in a high-level 3GL.  
<https://powcoder.com>
  - This produces an impedance mismatch - mixing different programming paradigms.
  - Estimated that as much as 30% of programming effort and code space is expended on this type of conversion.

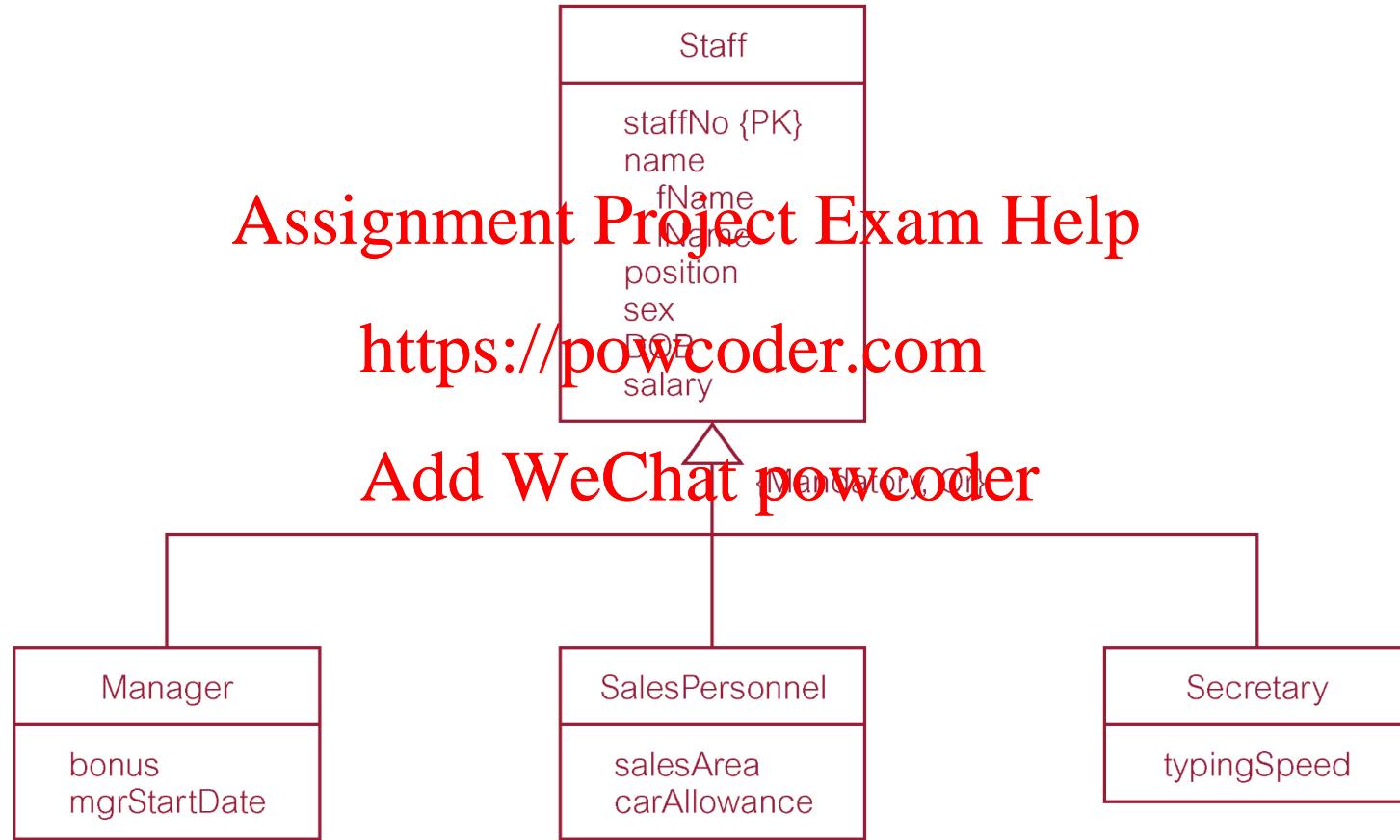
# Weaknesses of RDBMSs (5 of 5)

- Other Problems with RDBMSs
  - Transactions are generally short-lived and concurrent control protocols not suited for long-lived transactions.
  - Schema changes are difficult.
  - RDBMSs are AddWeChat [powcoder](https://powcoder.com).

# Storing Objects in Relational Databases (1 of 2)

- One approach to achieving persistence with an OOPL is to use an RDBMS as the underlying storage engine.
- Requires mapping class instances (i.e. objects) to one or more tuples distributed over one or more relations.  
<https://powcoder.com>
- To handle class hierarchy, have two basics tasks to perform:  
[Add WeChat powcoder](#)
  - (1) design relations to represent class hierarchy;
  - (2) design how objects will be accessed.

# Storing Objects in Relational Databases (2 of 2)



# Mapping Classes to Relations (1 of 2)

Number of strategies for mapping classes to relations, although each results in a loss of semantic information.

(1) Map each class or subclass to a relation.

Staff (**staffNo**, fName, lName, position, sex, DOB, salary) <https://powcoder.com> Add WeChat powcoder

Manager (**staffNo**, bonus, mgrStartDate)

SalesPersonnel (**staffNo**, salesArea, carAllowance)

Secretary (**staffNo**, typingSpeed)

# Mapping Classes to Relations (2 of 2)

(2) Map each subclass to a relation

Manager (**staffNo**, fName, lName, position, sex, DOB, salary, bonus, mgrStartDate)

Assignment Project Exam Help

SalesPersonnel (**staffNo**, fName, lName, position, sex, DOB, salary, salesArea, carAllowance)

Secretary (**staffNo**, fName, lName, position, sex, DOB, salary, typingSpeed)

Add WeChat powcoder

(3) Map the hierarchy to a single relation

Staff (**staffNo**, fName, lName, position, sex, DOB, salary, bonus, mgrStartDate, salesArea, carAllowance, typingSpeed, typeFlag)

# ORDBMSs

- RDBMSs currently dominant database technology with estimated sales of US\$24 billion in 2011, expected to grow to US\$37 billion by 2016.

**Assignment Project Exam Help**

- Vendors of RDBMSs conscious of threat and promise of OODBMS.
- Agree that RDBMSs not currently suited to advanced database applications, and added functionality is required.  
**Add WeChat powcoder**
- Reject claim that extended RDBMSs will not provide sufficient functionality or will be too slow to cope adequately with new complexity.
- Can remedy shortcomings of relational model by extending model with OO features.

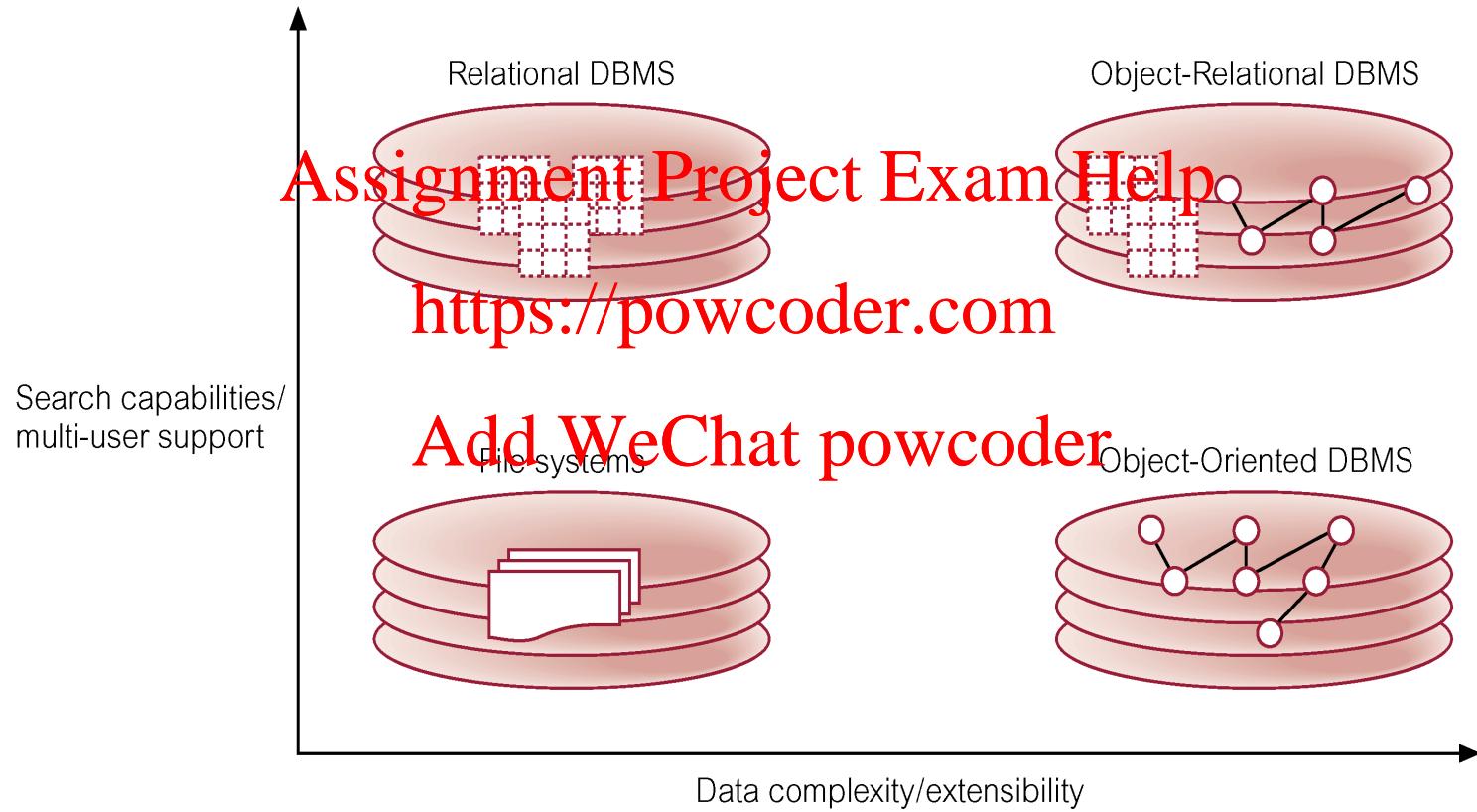
# ORDBMSs – Features (1 of 2)

- OO features being added include:
  - user-extensible types,
  - encapsulation, [Assignment Project Exam Help](#)
  - inheritance, <https://powcoder.com>
  - polymorphism,
  - dynamic binding of methods, [Add WeChat powcoder](#)
  - complex objects including non-1NF objects,
  - object identity.

## ORDBMSs – Features (2 of 2)

- However, no single extended relational model.
- All models:
  - share basic relational tables and query language,
  - all have some concept of ‘object’
  - some can store methods (or procedures or triggers).
- Some analysts predict ORDBMS will have 50% larger share of market than RDBMS.

# Stonebraker's View



# Advantages of ORDBMSs

- Resolves many of known weaknesses of RDBMS.
- Reuse and sharing:
  - reuse comes from ability to extend server to perform standard functionality centrally:  
<https://powcoder.com>
  - gives rise to increased productivity both for developer and end-user.  
[Add WeChat powcoder](#)
- Preserves significant body of knowledge and experience gone into developing relational applications.

# Disadvantages of ORDBMSs

- Complexity.
- Increased costs.  
**Assignment Project Exam Help**  
**https://powcoder.com**
- Proponents of relational approach believe simplicity and purity of relational model are lost.
- Some believe RDBMS is being extended for what will be a minority of applications.  
**Add WeChat powcoder**
- OO purists not attracted by extensions either.
- SQL now extremely complex.

# SQL:2011 - New OO Features

- Type constructors for row types and reference types.
- User-defined types (distinct types and structured types) that can participate in supertype/subtype relationships.
- User-defined procedures, functions, methods, and operators.  
<https://powcoder.com>  
Add WeChat powcoder
- Type constructors for collection types (arrays, sets, lists, and multisets).
- Support for large objects – BLOBS and CLOBs.
- Recursion.

# Row Types

- Sequence of field name/data type pairs that provides data type to represent types of rows in tables.
- Allows complete rows to be:
  - stored in variables, <https://powcoder.com>
  - passed as arguments to routines,
  - returned as return values from function calls.
- Also allows column of table to contain row values.

## Example 9.1 Use of Row Type

```
CREATE TABLE Branch (branchNo CHAR(4),
address ROW(street VARCHAR(25),
city VARCHAR(15),
postcode ROW(cityIdentifier VARCHAR(4),
subPart VARCHAR(4))));
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

```
INSERT INTO Branch
```

```
VALUES ('B005', ('22 Deer Rd', 'London',
ROW('SW1', '4EH')));
```

# User-Defined Types (UDTs) (1 of 2)

- SQL:2011 allows definition of UDTs.
- May be used in same way as built-in types.  
**Assignment Project Exam Help**
- Subdivided into two categories: distinct types and structured types.  
**https://powcoder.com**
- Distinct type allows differentiation between same underlying base types:

```
CREATE TYPE OwnerNoType AS VARCHAR(5)
FINAL;
```

```
CREATE TYPE StaffNoType AS VARCHAR(5) FINAL;
```

# User-Defined Types (UDTs) (2 of 2)

- Would get error if attempt to treat instance of one type as instance of other type.
- Not same as ~~Assignment Project Exam Help~~ SQL domains, which contains set of valid values that can be stored.  
<https://powcoder.com>
- Generally, UDT definition consists of one or more attribute definitions.  
[Add WeChat powcoder](#)
- Definition also consists of routine declarations (operator declarations deferred).
- Can also define equality and ordering relationships using CREATE ORDERING FOR.

# UDTs – Encapsulation and get/set Functions

- Value of an attribute can be accessed using common dot notation:

Assignment Project Exam Help

```
p.fName      p.fName = 'A. Smith'
```

- SQL encapsulates each attribute through an observer (get) and a mutator (set) function.
- These functions can be redefined by user in UDT definition.

```
FUNCTION fName(p PersonType) RETURNS VARCHAR(15)
RETURN p.fName;
```

# UDTs – Constructors and NEW Expression

- (Public) constructor function is automatically defined to create new instances of type:

Assignment Project Exam Help  
SET p = NEW PersonType();

- The constructor function has same name as type, takes 0 arguments, and returns a new instance with attributes set to their default values.
- User-defined constructor methods can be provided to initialize new instances. Must have same name as UDT but different parameters to public constructor

# UDTs - Example Constructor Method

```
CREATE CONSTRUCTOR METHOD PersonType (
    fN VARCHAR(15), IN VARCHAR(15), sx CHAR)
RETURNS PersonType SELF AS RESULT
BEGIN
    SET SELF.fName = fN;
    SET SELF.lName = IN;
    SET SELF.sex = sx;
    RETURN SELF;
END;
SET p = NEW PersonType('John', 'White' 'M');
```

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

Add WeChat powcoder

Add WeChat powcoder

Add WeChat powcoder

## Example 9.2 - Definition of New UDT (1 of 2)

```
CREATE TYPE PersonType AS (
    dateOfBirth DATE,
    fName      VARCHAR(15),
    lName      VARCHAR(15),
    sex        CHAR)
INSTANTIABLE
NOT FINAL
REF IS SYSTEM GENERATED
INSTANCE METHOD age() RETURNS INTEGER,
INSTANCE   METHOD   age(DOB   DATE)   RETURNS
PersonType;
```

## Example 9.2 - Definition of New UDT (2 of 2)

```
CREATE INSTANCE METHOD age () RETURNS INTEGER  
    FOR PersonType
```

```
BEGIN      Assignment Project Exam Help  
    RETURN    /* set age from SELF.dateOfBirth*/  
END;
```

<https://powcoder.com>

```
CREATE INSTANCE METHOD age(DOB DATE) RETURNS PersonType  
    FOR PersonType Add WeChat powcoder
```

```
BEGIN  
    SELF.dateOfBirth = /* set dateOfBirth from DOB*/  
    RETURN SELF;  
END;
```

# Subtypes and Supertypes

- UDTs can participate in subtype/supertype hierarchy using UNDER clause.
- Multiple inheritance is not supported.
- Subtype inherits all the attributes and behavior of its supertypes.  
<https://powcoder.com>  
Add WeChat powcoder
- Can define additional attributes and methods and can override inherited methods.
- Concept of substitutability supported: whenever instance of supertype expected instance of subtype can be used in its place.

## Example 9.3 - Creation of Subtype (1 of 2)

```
CREATE TYPE StaffType UNDER PersonType AS (
    staffNo  VARCHAR(5),  

    position  VARCHAR(10)      DEFAULT 'Assistant',
    salary    DECIMAL(7, 2),
    branchNo  CHAR(4))  

INSTANTIABLE  

NOT FINAL  

INSTANCE METHOD isManager() RETURNS BOOLEAN;
```

## Example 9.3 - Creation of Subtype (2 of 2)

```
CREATE INSTANCE METHOD isManager ()  
    RETURNS BOOLEAN FOR StaffType  
    BEGIN  
        IF SELF.position = 'Manager' THEN  
            RETURN TRUE;  
        ELSE  
            RETURN FALSE;  
        END IF  
    END)
```

Assignment Project Exam Help  
Add WeChat powcoder

# User-Defined Routines (UDRs) (1 of 4)

- UDRs define methods for manipulating data.
- UDRs may be defined as part of a UDT or separately as part of a schema.  
**Assignment Project Exam Help**
- An SQL-invoked routine may be a procedure, function, or method.  
**Add WeChat powcoder**
- May be externally provided in standard programming language or defined completely in SQL.

# User-Defined Routines (UDRs) (2 of 4)

- An SQL-invoked procedure is invoked from SQL CALL statement.
- May have zero or more parameters, each of which may be IN, OUT, or INOUT, and a body if defined fully within SQL.  
<https://powcoder.com>
- An SQL-invoked function returns a value.
- Any specified parameters must be input parameters with one designated as result parameter (using RESULT keyword).

# User-Defined Routines (UDRs) (3 of 4)

- SQL-invoked method is similar to a function but:
  - method is associated with a single UDT;
  - signature of every method of a UDT must be specified in UDT and definition of method must specify UDT.  
<https://powcoder.com>
- Three types of methods:
  - constructor methods, invoked using NEW;
  - instance methods, invoked using dot notation or using generalized invocation format;  
e.g. p.fName or (p AS StaffType).fName();
- static methods (analogous to class methods), invoked using ::; e.g. StaffType :: totalStaff().

# User-Defined Routines (UDRs) (4 of 4)

- External routine defined by specifying an external clause that identifies ‘compiled code’ in operating system’s file storage. [Assignment](#) [Project](#) [Exam](#) [Help](#)
- ORDBMS will provide method to dynamically link this object file into the DBMS so that it can be invoked when required. <https://powcoder.com> [Add WeChat](#) [powcoder](#)
- Procedure for this is outside bounds of SQL standard and is left as implementation-defined.

# Polymorphism

- Routine names may be overloaded, provided:
  - no two functions in same schema have same signature;
  - no two procedures in same schema have same name and number of parameters.
- Overriding applies only to methods and only based on runtime value of SELF argument
- SQL:2011 uses generalized object model, so types of all arguments considered when deciding which routine to invoke (left to right).
- Precedence lists used to determine closest match.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Reference Types and Object Identity (1 of 2)

- In SQL:2011, reference types can be used to define relationships between row types and uniquely identify a row within a table.
- Reference type value can be stored in one table and used as a direct reference to a specific row in some base table defined to be of this type (similar to pointer type in ‘C’/C++).
- In this way, reference type provides similar functionality as OID of OODBMSs.

# Reference Types and Object Identity (2 of 2)

- Thus, references allow a row to be shared among multiple tables, and enable users to replace complex join definitions in queries with much simpler path expressions.
- References also give optimizer alternative way to navigate data instead of using value-based joins.
- REF IS SYSTEM GENERATED in CREATE TYPE indicates that actual values of associated REF type are provided by the system.

## Example 9.4 Table Creation based on UDT

```
CREATE TABLE Person (
    info    PersonType,
    CONSTRAINT DOB_Check
        CHECK(dateOfBirth > DATE'1900-01-01'));
https://powcoder.com
```

- or

```
Add WeChat powcoder
CREATE TABLE Person OF PersonType (
    dateOfBirth WITH OPTIONS
    CONSTRAINT DOB_Check
        CHECK(dateOfBirth > DATE'1900-01-01')
    REF IS personID SYSTEM GENERATED);
```

# Subtables and Supertables

- No mechanism to store all instances of given UDT, unless user explicitly creates a single table in which all instances are stored  
**Assignment Project Exam Help**
- Thus, in SQL:2011 may not be possible to apply an SQL query to all instances of a given UDT.  
**https://powcoder.com**
- Can use table inheritance, which allows table to be created that inherits all the attributes of one or more existing tables using UNDER clause.  
**Add WeChat powcoder**
- Subtable/supertable independent from UDT inheritance facility.

## Example 9.5 Creation of Subtable

CREATE TABLE Staff OF StaffType UNDER Person;

- Each row of supertable Person can correspond to at most one row in Staff.
- Each row in Staff must have exactly one corresponding row in Person.
- Containment used: row of subtable ‘contained’ in one of its supertables.

## Example 9.6 Using Reference Type to Define a Relationship

```
CREATE TABLE PropertyForRent (
    propertyNo Property Number NOT NULL,
    street      Street          NOT NULL,
                https://powcoder.com
    ....
    staffID     REF(StaffType)  SCOPE Staff
    REFERENCES ARE CHECKED ON DELETE CASCADE,
    PRIMARY KEY (propertyNo);
```

## Example 9.7 Retrieve Specific Column/Rows

Find the names of all Managers.

```
SELECT s.IName  
FROM Staffs  
WHERE s.position = 'Manager';
```

- Uses implicitly defined observer function.

## Example 9.8 Invoke User-Defined Function

Find the names and ages of all Managers.

```
SELECT s.lName, s.age
```

```
FROM Staff
```

```
WHERE s.isManager;
```

- Uses user-defined function `isManager` as a predicate of the WHERE clause (returns TRUE if member of staff is a manager).
- Also uses inherited virtual observer function `age`.

## Example 9.9 Use of ONLY (1 of 2)

Find names of all people over 65.

```
SELECT p.lName, p.firstName  
FROM Person p  
WHERE p.age > 65;
```

- This will list out not only records explicitly inserted into Person table, but also records inserted directly/indirectly into subtables of Person.

## Example 9.9 Use of ONLY (2 of 2)

- Can restrict access to specific instances of Person table, excluding any subtables, using ONLY.

Assignment Project Exam Help

```
SELECT p.lName, p fName  
FROM ONLY (Person) p  
WHERE p.age > 65
```

<https://powcoder.com>

Add WeChat powcoder

# Example 9.10 Use of Dereference Operator (1 of 2)

Find name of member of staff who manages property PG4.

```
SELECT p.staffID→ fName AS fName,  
       Assignment Project Exam Help  
       p.staffID→ IName AS IName,  
FROM PropertyForRent p  
WHERE p.propertyNo = 'PG4';  
      Add WeChat powcoder
```

- In SQL2, this query would have required a join or nested subquery

## Example 9.10 Use of Dereference Operator (2 of 2)

To retrieve the member of staff for property PG4, rather than just the first and last name:

```
SELECT DEREF(p.staffID) AS Staff  
FROM PropertyForRent p  
WHERE p.propertyNo = 'PG4';
```

- Note, reference types by themselves do not provide referential integrity.

# Collection Types (1 of 2)

- Collections are type constructors used to define collections of other types.
- Used to store multiple values in a single column and can result in nested tables.  
<https://powcoder.com>
- SQL:2011 has parameterized ARRAY and MULTISET collection types. [Add WeChat powcoder](#)
- Later SQL may have parameterized LIST and SET collection types.
- The parameter may be predefined type, UDT, row type, or another collection (but not reference type or UDT containing reference type).

## Collection Types (2 of 2)

**ARRAY:** 1D array with maximum number of elements.

**LIST:** ordered collection that allows duplicates.

[Assignment](#) [Project](#) [Exam](#) [Help](#)

**SET:** unordered collection that does not allow duplicates.

<https://powcoder.com>

**MULTISET:** unordered collection that allows duplicates.

- Similar to those in the ODMG 3.0 standard.

[Add WeChat](#) [powcoder](#)

## Example 9.11 Use of ARRAY Collection

- Branch has up to three telephone numbers:

telNo VARCHAR(13) ARRAY[3]  
**Assignment Project Exam Help**

Retrieve first and last phone number for B003.

<https://powcoder.com>

```
SELECT telNo[1], telNo[CARDINALITY(telNo)]  
FROM Branch  
WHERE branchNo = 'B003';  
  
Add WeChat powcoder
```

# Multiset

- Unordered collection of elements, all of same type, with duplicates permitted.
- Since multiset is unordered there is no ordinal position to reference individual elements. Unlike arrays, multiset is an unbounded collection.
- Analogous to tables, operators are provided to convert multiset to table (UNNEST) and table to multiset (MULTISET).

# Operations on MULTISET (1 of 2)

- SET, removes duplicates from a multiset to produce a set.
- CARDINALITY, returns number of current elements.
- ELEMENT, returns element of a multiset if multiset only has one element (or null if multiset has no elements).  
Exception raised if multiset has more than one element.

# Operations on MULTISET (2 of 2)

- MULTISET UNION, computes union of two multisets; keywords ALL or DISTINCT can retain duplicates or remove them [Assignment Project Exam Help](https://powcoder.com)
- MULTISET INTERSECT, computes intersection of two multisets; keyword DISTINCT can remove duplicates; keyword ALL can place in result as many instances of each value as minimum number of instances of that value in either operand. [Add WeChat powcoder](https://powcoder.com)
- MULTISET EXCEPT, computes difference of two multisets; again, keyword DISTINCT or ALL can be specified.

# Aggregate Functions for MULTISET

- COLLECT, creates multiset from value of the argument in each row of a group;
- FUSION, creates multiset union of a multiset value in all rows of a group; [Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- INTERSECTION, creates multiset intersection of a multiset value in all rows of a group. [Add WeChat powcoder](#)

# Predicates for use with MULTISET

- Comparison predicate (equality and inequality only);
- DISTINCT predicate;
- MEMBER predicate;
- SUBMULTISET predicate, which tests whether one multiset is a submultiset of another;
- IS A SET/IS NOT A SET predicate, which checks whether a multiset is a set.

# Example 9.12 Use of Collection MULTISET

- Extend Staff table to contain details of next-of-kin:

nok NameType MULTISET  
**Assignment Project Exam Help**

Find first and last names of John White's next of kin.

<https://powcoder.com>  
SELECT n.fName, n.lName  
FROM Staff s,  
UNNEST(n.nok) AS n(fName, lName)  
WHERE s.lName = 'White' AND s.fName = 'John';  
Add WeChat powcoder

# Example 9.13 FUSION and INTERSECTION (1 of 2)

propertyNo	viewDates
PA14	MULTISET['16-Apr-13', '18-Apr-13', '20-May-13'] Assignment Project Exam Help
PG4	MULTISET['20-Apr-13', '14-May-13', '26-May-13'] <a href="https://powcoder.com">https://powcoder.com</a>
PG36	MULTISET['28-Apr-13', '14-May-13'] Add WeChat powcoder
PL94	Null

```
SELECT FUSION(viewDates) AS viewDateFusion,  
        INTERSECTION(viewDates) AS viewDateIntersection  
FROM PropertyViewDates;
```

# Example 9.13 FUSION and INTERSECTION (2 of 2)

`viewDateFusion`

`MULTISET['14-May-13', '14-May-13',  
'14-May-13', '24-May-13', '20-Apr-13',  
'26-May-13', '28-Apr-13']`

`viewDateIntersection`

`MULTISET['14-May-13']`

**Assignment Project Exam Help**

<https://powcoder.com>

Add WeChat powcoder

## Example 9.14 Typed Views

```
CREATE VIEW FemaleView OF
    PersonType (REF IS personID DERIVED)
AS SELECT fName, lName
FROM ONLY (Person)
WHERE sex = 'F';
```

```
CREATE VIEW FemaleStaffView OF
    StaffType UNDER FemaleView
AS SELECT fName, lName, staffNo, position
FROM ONLY (Staff)
WHERE branchNo = 'B003';
```

# Persistent Stored Modules (SQL/PSM) (1 of 2)

- SQL:2011 has some new statement types to make it computationally complete.
- Behavior (methods) can be stored and executed from within database as SQL statements.  
<https://powcoder.com>
- Can group statements into a compound statement (block), with its own local variables.

# Persistent Stored Modules (SQL/PSM) (2 of 2)

- Some of the new statements are:
  - An assignment statement.
  - An IF ... THEN ... ELSE ... END IF statement.
  - CASE statement. <https://powcoder.com>
  - A set of statements for iteration: FOR, WHILE, and REPEAT. [Add WeChat powcoder](#)
  - A CALL statement to invoke procedures and a RETURN statement.
  - Condition handling.

# Triggers (1 of 4)

- An SQL (compound) statement executed automatically by DBMS as side effect of a modification to named table.
- Use of triggers include:
  - Validating input data and maintaining complex integrity constraints that otherwise would be difficult/impossible.
  - Supporting alerts.
  - Maintaining audit information.
  - Supporting replication.

# Triggers (2 of 4)

CREATE TRIGGER TriggerName

BEFORE | AFTER <triggerEvent>  
**Assignment Project Exam Help**

ON <TableName>

**https://powcoder.com**

[REFERENCING <oldOrNewValuesAliasList>]

**Add WeChat powcoder**  
[FOR EACH {ROW | STATEMENT}]

[ WHEN (triggerCondition) ]

<triggerBody>

# Triggers (3 of 4)

- BEFORE trigger fired before and AFTER trigger is fired after associated event occurs.
- Triggered action is SQL procedure statement, which can be executed in one of two ways:
  - For each row (FOR EACH ROW) affected by the event. This is called a row-level trigger.
  - Only once for entire event (FOR EACH STATEMENT), which is default. This is called a statement-level trigger.

# Triggers (4 of 4)

- As more than one trigger can be defined on a table, order of firing is important. The following order is observed:

(1) Execution of any BEFORE triggers on table.

(2) For each row affected by the statement:

- Execute any BEFORE row-level trigger.
- Execute the statement itself.
- Apply any referential constraints.
- Execute any AFTER row-level trigger.

(3) Execute any AFTER trigger on table.

## Example 9.15 Use of AFTER Trigger

```
CREATE TRIGGER InsertMailshotTable
    AFTER INSERT ON PropertyForRent
    REFERENCING NEW ROW AS pfr
    BEGIN
        INSERT INTO Mailshot VALUES
        (SELECT c.fName, c.lName, c.maxRent,
        pfr.propertyNo, pfr.street, pfr.city, pfr.postcode,
        pfr.type, pfr.rooms, pfr.rent
        FROM Client c
        WHERE c.branchNo = pfr.branchNo AND
        (c.prefType=pfr.type AND c.maxRent <= pfr.rent) )
    END;
```

## Example 9.16 Use of AFTER Trigger with Condition

```
CREATE TRIGGER UpdateMailshotTable
    AFTER UPDATE OF rent ON PropertyForRent
    REFERENCING NEW ROW AS pfr
    FOR EACH ROW
    BEGIN ATOMIC
        DELETE FROM Mailshot WHERE maxRent > pfr.rent;
        UPDATE Mailshot SET rent = pfr.rent
        WHERE propertyNo = pfr.propertyNo;
    END;
```

# Triggers - Advantages and Disadvantages

- Major advantage - standard functions can be stored within database and enforced consistently.
- This can dramatically reduce complexity of applications.
- However, there can be some disadvantages:
  - Complexity.
  - Hidden functionality.
  - Performance overhead.

# Large Objects

- A table field that holds large amount of data.
- Three different types:
  - Binary Large Object (BLOB).
  - Character LOB (CLOB) and National CLOB.
- SQL:2011 LOB slightly different from original type of BLOB that appears in many current DBMSs, where BLOB is non-interpreted byte stream.
- In SQL:2011, LOB does allow some operations to be carried out in DBMS server.

## Example 9.17 Use of CLOB and BLOB

- Extend Staff table to hold a resume and picture for the staff member.

Assignment Project Exam Help

```
ALTER TABLE Staff  
    ADD COLUMN resume    CLOB(50K);  
  
ALTER TABLE Staff  
    ADD COLUMN picture   BLOB(12M);
```

# Object-Oriented Extensions in Oracle (1 of 3)

- Many of the object-oriented features that appear in new SQL:2011 standard appear in Oracle in one form or another. [Assignment Project Exam Help](https://powcoder.com)
- Oracle supports two user-defined data types:
  - object types;
  - collection types.<https://powcoder.com>  
[Add WeChat powcoder](#)

# Object Types in Oracle (1 of 2)

- An object type is a schema object that has a name, a set of attributes based on the Oracle built-in data types or possibly other object types, and a set of methods.

**Assignment Project Exam Help**

<https://powcoder.com>  
CREATE TYPE Address Type AS OBJECT (  
 street VARCHAR2(25),  
 city VARCHAR2(15),  
 postcode VARCHAR2(8));

# Object-Oriented Extensions in Oracle (2 of 3)

```
CREATE TYPE StaffType AS OBJECT (
    staffNo    VARCHAR2(5),
    fName      VARCHAR2(15),
    ....
    MAP MEMBER FUNCTION age
        Add WeChat powcoder
        RETURN INTEGER,
PRAGMA RESTRICT_REFERENCES(
    age, WNDS, WNPS, RNPS));
```

# Object-Oriented Extensions in Oracle (3 of 3)

```
CREATE TYPE BranchType AS OBJECT (
    branchNo VARCHAR2(4),
    address AddressType,
    MAP MEMBER FUNCTION getbranchNo
        Add WeChat powcoder
        RETURN VARCHAR2(4),
    PRAGMA RESTRICT_REFERENCES(
        getbranchNo, WNDS, WNPS, RNDS, RNPS));
```

# Object Types in Oracle (2 of 2)

- Pragma clause is a compiler directive that denies member functions read/write access to database tables and/or package variables.
- Can now create a Branch (Object) table:  
<https://powcoder.com>

CREATE TABLE Branch OF BranchType  
(branchNo PRIMARY KEY);

# Methods in Oracle (1 of 3)

- Methods of an object type are classified as member, static, and comparison.
- Member method is a function/procedure that always has implicit SELF parameter as first parameter (whose type is containing object type):
  - Useful as observer and mutator functions.
- Static method is a function/procedure that does not have an implicit SELF parameter.
  - Useful for specifying user-defined constructors or cast methods and may be invoked by qualifying method with the type name, as in typename.method().

## Methods in Oracle (2 of 3)

- Comparison method used for comparing instances of objects.
- Oracle provides two ways to define an order relationship among objects of a given type:  
<https://powcoder.com>
  - a map method uses Oracle's ability to compare built-in types. [Add WeChat powcoder](#)
  - an order method uses its own internal logic to compare two objects of a given object type. It returns a value that encodes the order relationship. For example, may return  $-1$  if first is smaller,  $0$  if they are equal, and  $1$  if first is larger.

# Methods in Oracle (3 of 3)

- Methods can be implemented in PL/SQL, Java, and ‘C’.
- Overloading is supported provided their formal parameters differ in number, order, or data type.

[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

# Object Identifiers

- Every row object in an object table has associated logical OID, which uniquely identifies the row.
- The OID column is hidden from users and there is no access to its internal structure.  
<https://powcoder.com>
- Oracle requires every row object to have a unique OID, which may be specified to come from the row object's PK or to be system-generated.

```
CREATE TABLE Branch OF BranchType  
    (branch N o PRIMARYKEY)  
    OBJECT IDENTIFIER PRIMARYKEY;
```

# REF Data Type (1 of 2)

- Oracle provides a built-in data type called REF to encapsulate references to row objects of a specified object type.  
[Assignment Project Exam Help](https://powcoder.com)
- In effect, a REF is used to model an association between two row objects.
- A REF can be used to examine or update the object it refers to and to obtain a copy of the object it refers to.
- Only changes that can be made to a REF are to replace its contents with a reference to a different object of same object type or to assign it a null value.

## REF Data Type (2 of 2)

```
CREATE TYPE BranchType AS OBJECT (
    branchNo           VARCHAR2(4),
    address            AddressType,
    manager             REF StaffType,
)
MAP MEMBER FUNCTION
    getbranchNo RETURN VARCHAR2(4),
PRAGMA RESTRICT_REFERENCES(
    getbranchNo, WNDS, WNPS, RNDS, RNPS);
```

# Collection Types

- Oracle supports two collection types: **array types** and **table types**.
- An array is an ordered set of data elements, all of same data type. <https://powcoder.com>
- Each element has an **index**, a number corresponding to the element's position in the array.
- An array can have a fixed or variable size, although in latter case maximum size must be specified when array type is declared.

# Nested Tables (1 of 3)

- An unordered set of data elements, all of same data type.
- It has a single column of a built-in type or an object type.  
**Assignment Project Exam Help**
- If column is an object type, table can also be viewed as a multi-column table, with a column for each attribute of the object type.  
**https://powcoder.com**

Add WeChat powcoder

## Nested Tables (2 of 3)

```
CREATE TYPE NextOfKinType AS OBJECT (
    fName  VARCHAR2(15),
    lName  VARCHAR2(15),
    telNo  VARCHAR2(13));
```

```
CREATE TYPE NextOfKinNestedType AS
TABLE OF NextOfKinType;
```

- Can now modify StaffType to include this new type:

```
nextOfKin NextOfKinNestedType
```

## Nested Tables (3 of 3)

- Can now create Staff table:

```
CREATE TABLE Staff OF StaffType(  
    PRIMARY KEY staffNo)  
    OBJECT IDENTIFIER PRIMARY KEY  
    NESTED TABLE nextOfKin STORE AS NextOfKin  
    StorageTable(Add WeChat powcoder  
        (PRIMARY KEY(Nested_Table_Id, lName, telNo))  
        ORGANIZATION INDEX COMPRESS)  
    RETURN AS LOCATOR;
```

# Manipulating Object Tables (1 of 2)

```
INSERT INTO Staff VALUES ('SG37', 'Ann',  
    'Beech', 'Assistant', 'F', '10-Nov-1960', 12000,
```

Assignment Project Exam Help  
NextOfKinNestedType());

```
INSERT INTO TABLE (SELECT nextOfKin  
    FROM Staff s  
    WHERE s.staffNo = 'SG5')  
VALUES ('John', 'Brand', '0141-848-2000');
```

# Manipulating Object Tables (2 of 2)

- Can now insert object into Branch table:

Assignment Project Exam Help  
INSERT INTO Branch

SELECT 'B003', AddressType('163 Main St',  
'Glasgow', 'G11 9QX'), REF(s),  
TelNoArrayType('0141-339-2178',  
'0141-339-4439')  
FROM Staff s  
WHERE s.staffNo = 'SG5';

# Querying Object Tables

```
SELECT b.branchNo  
FROM Branch b  
ORDER BY VALUE(b);
```

```
Assignment Project Exam Help  
SELECT b.branchNo, b.address,  
       DEREF(b.manager), b.phoneList  
FROM Branch b  
WHERE b.address.city = 'Glasgow'  
ORDER BY VALUE(b);
```

# Object Views

- **Object view** is a virtual object table.
- In Oracle, can create an object view that not only restricts access to some data but also prevents some methods from being invoked, such as a delete method.  
<https://powcoder.com>
- Also argued that object views provide a simple migration path from a purely relational-based application to an object-oriented one, thereby allowing companies to experiment with this new technology.

# Copyright



This work is protected by United States copyright laws and is provided solely for the use of instructors in teaching their courses and assessing student learning. Dissemination or sale of any part of this work (including on the World Wide Web) will destroy the integrity of the work and is not permitted. The work and materials from it should never be made available to students except by instructors using the accompanying text in their classes. All recipients of this work are expected to abide by these restrictions and to honor the intended pedagogical purposes and the needs of other instructors who rely on these materials.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Database Systems: A Practical Approach to Design, Implementation, and Management

Sixth Edition



Assignment Project Exam Help Chapter 27

<https://powcoder.com>  
Object-Oriented DBMSs –  
Concepts and Design

Add WeChat powcoder

# Learning Objectives (1 of 2)

**27.1** The next generation of database systems.

**27.2** Framework for an OO data model.

**27.3** Basics of the FDM.

**27.4** Basics of persistent programming languages.

**27.5** Main strategies for developing an OODBMS.

**27.6** Single-level v. two-level storage models.

**27.7** Pointer swizzling.

**27.8** How an OODBMS accesses records.

**27.8** Persistent schemes.

# Learning Objectives (2 of 2)

**27.9** Advantages and disadvantages of orthogonal persistence.

**27.9** Issues underlying OODBMSs. Assignment Project Exam Help

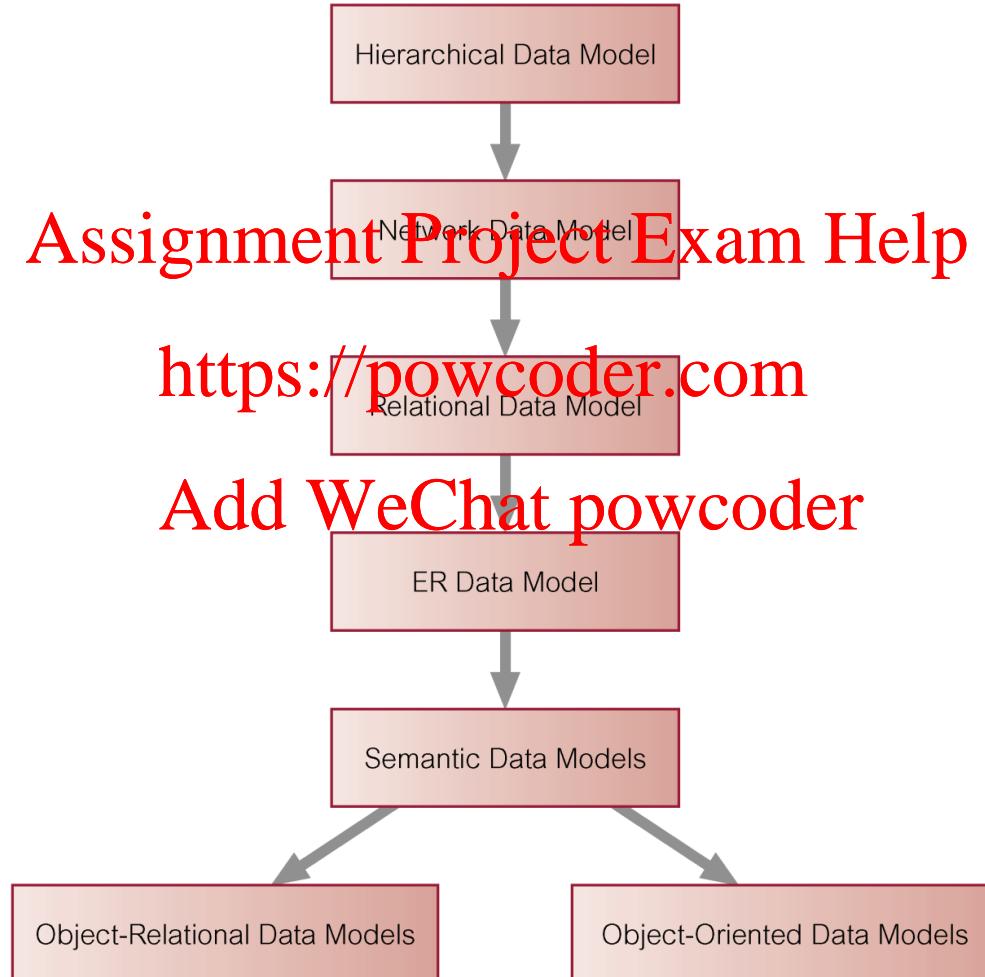
**27.10** Advantages and disadvantages of OODBMSs.

Add WeChat powcoder

# Next Generation Database Systems

- **First Generation DBMS:** Network and Hierarchical
  - Required complex programs for even simple queries.
  - Minimal data independence.
  - No widely accepted theoretical foundation.  
<https://powcoder.com>
- **Second Generation DBMS:** Relational DBMS
  - Helped overcome these problems.  
[Add WeChat powcoder](#)
- **Third Generation DBMS:** OODBMS and ORDBMS.

# History of Data Models



# Object-Oriented Data Model (1 of 3)

**No one agreed object data model. One definition:**

- Object-Oriented Data Model (OODM)
  - Data model that captures semantics of objects supported in object-oriented programming.  
<https://powcoder.com>
- Object-Oriented Database (OODB)
  - Persistent and sharable collection of objects defined by an ODM.
- Object-Oriented DBMS (OODBMS)
  - Manager of an ODB.

# Object-Oriented Data Model (2 of 3)

- Zdonik and Maier present a threshold model that an OODBMS must, at a minimum, satisfy:
  - It must provide database functionality.
  - It must support object identity.
  - It must provide encapsulation.
  - It must support objects with complex state.

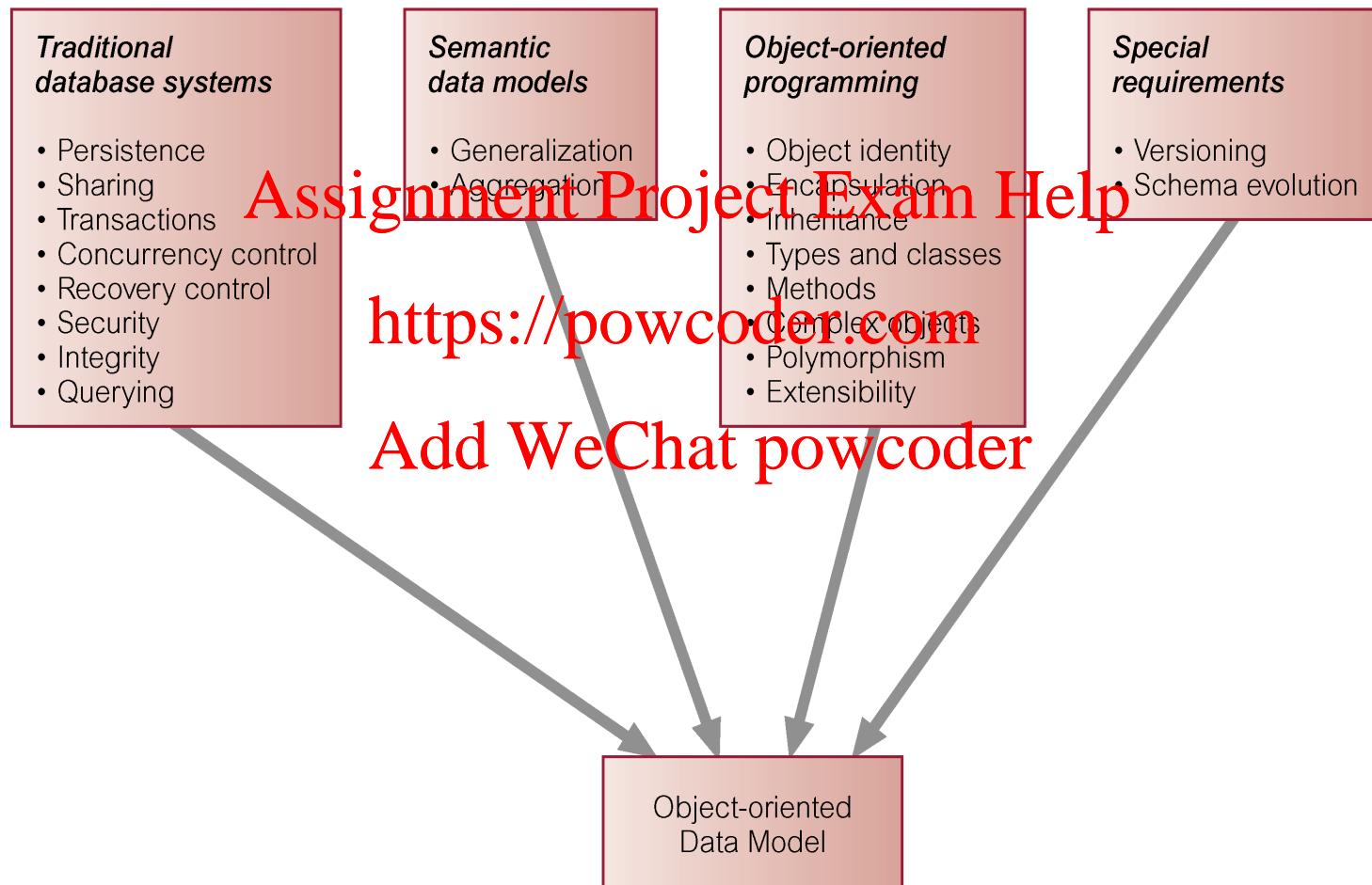
# Object-Oriented Data Model (3 of 3)

- Khoshafian and Abnous define OODBMS as:
  - OO = ADTs + Inheritance + Object identity
  - OODBMS = OO + Database capabilities.
- Parsaye et al. gives <https://powcoder.com>
  - High-level query language with query optimization.
  - Support for persistence, atomic transactions: concurrency and recovery control.
  - Support for complex object storage, indexes, and access methods.
  - OODBMS = OO system + (1), (2), and (3).

# Commercial OODBMSs

- GemStone/S from Gemstone Systems Inc.,
- Objectivity/DB from Objectivity Inc.,  
**Assignment Project Exam Help**
- ObjectStore from Progress Software Corp.,  
**https://powcoder.com**
- Versant Object Database db40 and FastObjects from  
Versant Corp. **Add WeChat powcoder**

# Origins of the Object-Oriented Data Model



# Functional Data Model (FDM)

- Interesting because it shares certain ideas with object approach including object identity, inheritance, overloading and navigational access
- In FDM, any data retrieval task can be viewed as process of evaluating and returning result of a function with zero, one, or more arguments.
- Resulting data model is conceptually simple but very expressive.
- In the FDM, the main modeling primitives are **entities** and **functional relationships**.

# FDM - Entities

- Decomposed into (abstract) entity types and printable entity types.
- Entity types correspond to classes of real world' objects and declared as functions with 0 arguments that return type ENTITY.  
**Assignment Project Exam Help**  
**<https://powcoder.com>**
- For example:  
Staff() → ENTITY  
PropertyForRent() → ENTITY.

# FDM – Printable Entity Types and Attributes

- Printable entity types are analogous to base types in a programming language.
- Include: INTEGER, CHARACTER, STRING, REAL, and DATE.  
[Assignment Project Exam Help](https://powcoder.com)  
[Add WeChat powcoder](#)
- An attribute is a **functional relationship**, taking the entity type as an argument and returning a printable entity type.
- For example:

staffNo(Staff) → STRING

sex(Staff) → CHAR

salary(Staff) → REAL

# FDM – Composite Attributes

Name() → ENTITY

Name(Staff) → NAME

Assignment Project Exam Help

fName(Name) → STRING

<https://powcoder.com>

IName(Name) → STRING

Add WeChat powcoder

# FDM – Relationships (1 of 2)

- Functions with arguments also model relationships between entity types.
- Thus, FDM makes no distinction between attributes and relationships. [Assignment Project Exam Help](https://powcoder.com) <https://powcoder.com> [Add WeChat powcoder](#)
- Each relationship may have an inverse relationship defined.
- For example:

Manages(Staff) → PropertyForRent

ManagedBy(PropertyForRent) → Staff INVERSE OF  
Manages

# FDM – Relationships (2 of 2)

- Can also model  $* : *$  relationships:
  - Views(Client) —» PropertyForRent
  - ViewedBy(PropertyForRent) —» Client INVERSE OF Views  
<https://powcoder.com>
- and attributes on relationships:
  - viewDate(Client, PropertyForRent) → DATE

# FDM – Inheritance and Path Expressions

- Inheritance supported through entity types.
- Principle of substitutability also supported.

Staff() → ENTITY

Supervisor() → ENTITY

<https://powcoder.com>

IS-A-STAFF(Supervisor) → Staff

- Derived functions can be defined from composition of multiple functions (note overloading):

fName(Staff) → fName(Name(Staff))

fName(Supervisor) → fName(IS-A-STAFF(Supervisor))

- Composition is a **path expression** (cf. dot notation):

# FDM – Declaration of FDM Schema

## Entity type declarations

Staff() → ENTITY  
Supervisor() → ENTITY

PropertyForRent() → ENTITY  
Client() → ENTITY

Name → ENTITY

## Attribute declarations

fName(Name) → STRING  
lName(Name) → STRING  
fName(Staff) → fName(Name(Staff))  
lName(Staff) → lName(Name(Staff))  
fName(Client) → fName(Name(Client))  
lName(Client) → lName(Name(Client))

staffNo(Staff) → STRING  
position(Staff) → STRING  
sex(Staff) → CHAR  
salary(Staff) → REAL  
clientNo(Client) → STRING  
telNo(Client) → STRING  
prefType(Client) → STRING  
maxRent(Client) → REAL

propertyNo(PropertyForRent) → STRING  
street(PropertyForRent) → STRING  
city(PropertyForRent) → STRING  
telePropertyForRent() → STRING  
rooms(PropertyForRent) → INTEGER  
rent(PropertyForRent) → REAL

Assignment Project Exam Help

<https://powcoder.com>

## Relationship type declarations

Manages(Staff) → PropertyForRent  
ManagedBy(PropertyForRent) → Staff INVERSE OF Manages  
Views(Client) → PropertyForRent  
ViewedBy(PropertyForRent) → Client INVERSE OF Views  
viewDate(Client, PropertyForRent) → DATE  
comments(Client, PropertyForRent) → STRING

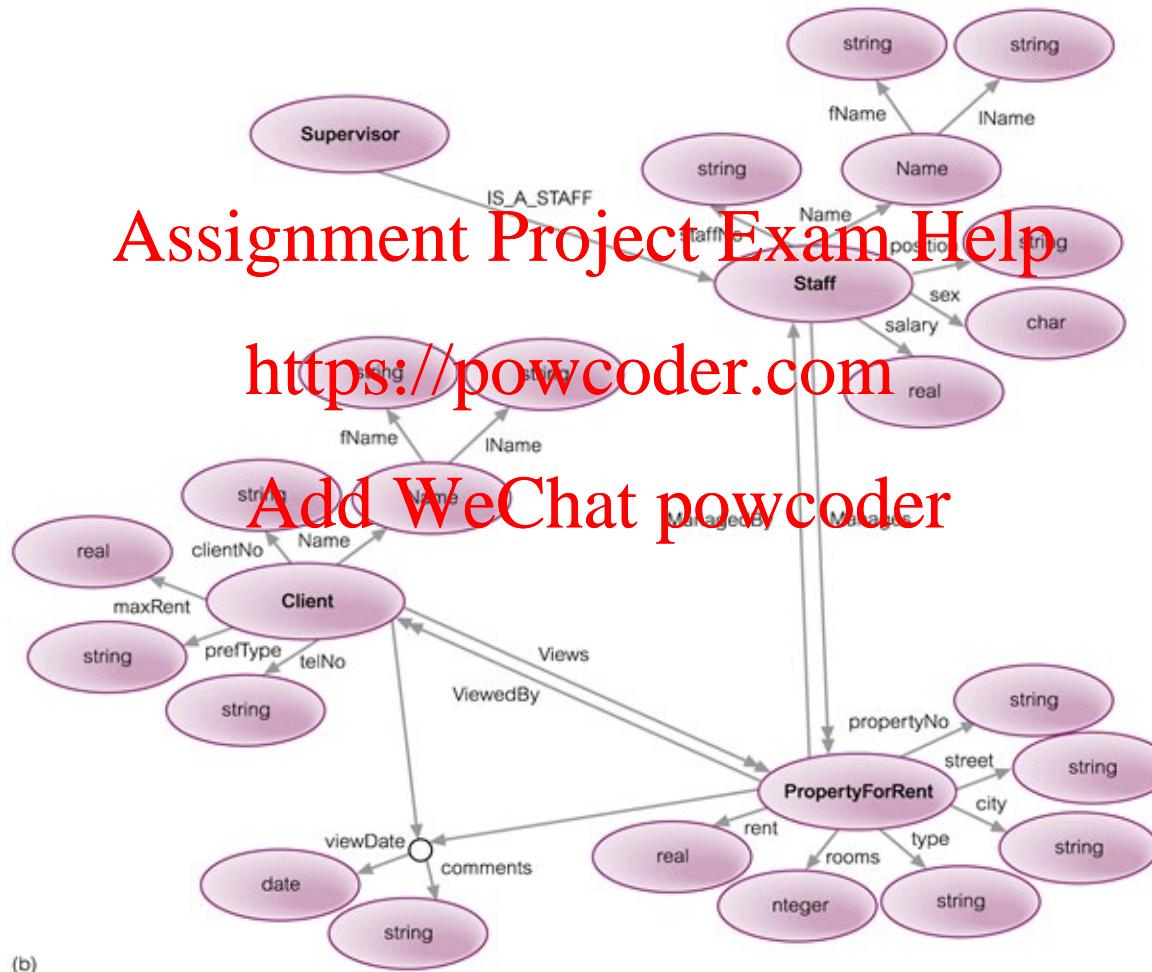
## Inheritance declarations

IS-A-STAFF(Supervisor) → Staff  
staffNo(Supervisor) → staffNo(IS-A-STAFF(Supervisor))  
fName(Supervisor) → fName(IS-A-STAFF(Supervisor))  
lName(Supervisor) → lName(IS-A-STAFF(Supervisor))  
position(Supervisor) → position(IS-A-STAFF(Supervisor))  
sex(Supervisor) → sex(IS-A-STAFF(Supervisor))  
salary(Supervisor) → salary(IS-A-STAFF(Supervisor))

(a)

(a) Declaration of part of *DreamHome* as an FDM schema; (b) corresponding diagrammatic representation.

# FDM – Diagrammatic Representation of Schema



(b)

# FDM – Functional Query Languages

- Path expressions also used within a functional query.
- For example:

Assignment Project Exam Help  
<https://powcoder.com>

```
RETRIEVE IName(Name(Viewed By(Manages(Staff))))  
WHERE staffNo(Staff) = 'SG14'
```

- or in dot notation:

Add WeChat powcoder  
<https://powcoder.com>

```
RETRIEVE Staff.Manages.Viewed By.Name.IName  
WHERE Staff.staffNo='SG14'
```

# FDM – Advantages

- Support for some object-oriented concepts.
- Support for referential integrity.

Assignment Project Exam Help

- Irreducibility.

<https://powcoder.com>

- Easy extensibility.

Add WeChat powcoder

- Suitability for schema integration.

- Declarative query language.

# Persistent Programming Languages (PPLs) (1 of 3)

- Language that provides users with ability to (transparently) preserve data across successive executions of a program and even allows such data to be used by many different programs.  
<https://powcoder.com>
- In contrast, database programming language (e.g. SQL) differs by its incorporation of features beyond persistence, such as transaction management, concurrency control, and recovery.

# Persistent Programming Languages (PPLs) (2 of 3)

- PPLs eliminate impedance mismatch by extending programming language with database capabilities.
  - In PPL, ~~Assignment Project Exam Help~~ language's type system provides data model, containing rich structuring mechanisms.  
<https://powcoder.com>
- In some PPLs procedures are ‘first class’ objects and are treated like any ~~Add WeChat powcoder~~ object in language.
  - Procedures are assignable, may be result of expressions, other procedures or blocks, and may be elements of constructor types.
  - Procedures can be used to implement ADTs.

# Persistent Programming Languages (PPLs) (3 of 3)

- PPL also maintains same data representation in memory as in persistent store.
  - Overcomes difficulty and overhead of mapping between the two representations.  
<https://powcoder.com>
- Addition of (transparent) persistence into a PPL is important enhancement to IDE, and integration of two paradigms provides more functionality and semantics.

# Alternative Strategies for Developing an OODBMS (1 of 2)

- Extend existing OO programming language.
  - GemStone extended Smalltalk.
- Provide extensible OODBMS library.
  - Approach taken by Ontos, Versant, and ObjectStore.
- Embed OODB language constructs in a conventional host language.
  - Approach taken by O<sub>2</sub>, which has extensions for C.

# Alternative Strategies for Developing an OODBMS (2 of 2)

- Extend existing database language with object-oriented capabilities.
  - Approach being pursued by RDBMS and OODBMS vendors.
  - Ontos and Versant provide a version of OSQL.
- Develop a novel database data model/language.

# Single-Level v. Two-Level Storage Model (1 of 3)

- Traditional programming languages lack built-in support for many database features.
- Increasing number of applications now require functionality from both database systems and programming languages.  
<https://powcoder.com>
- Such applications need to store and retrieve large amounts of shared, structured data.

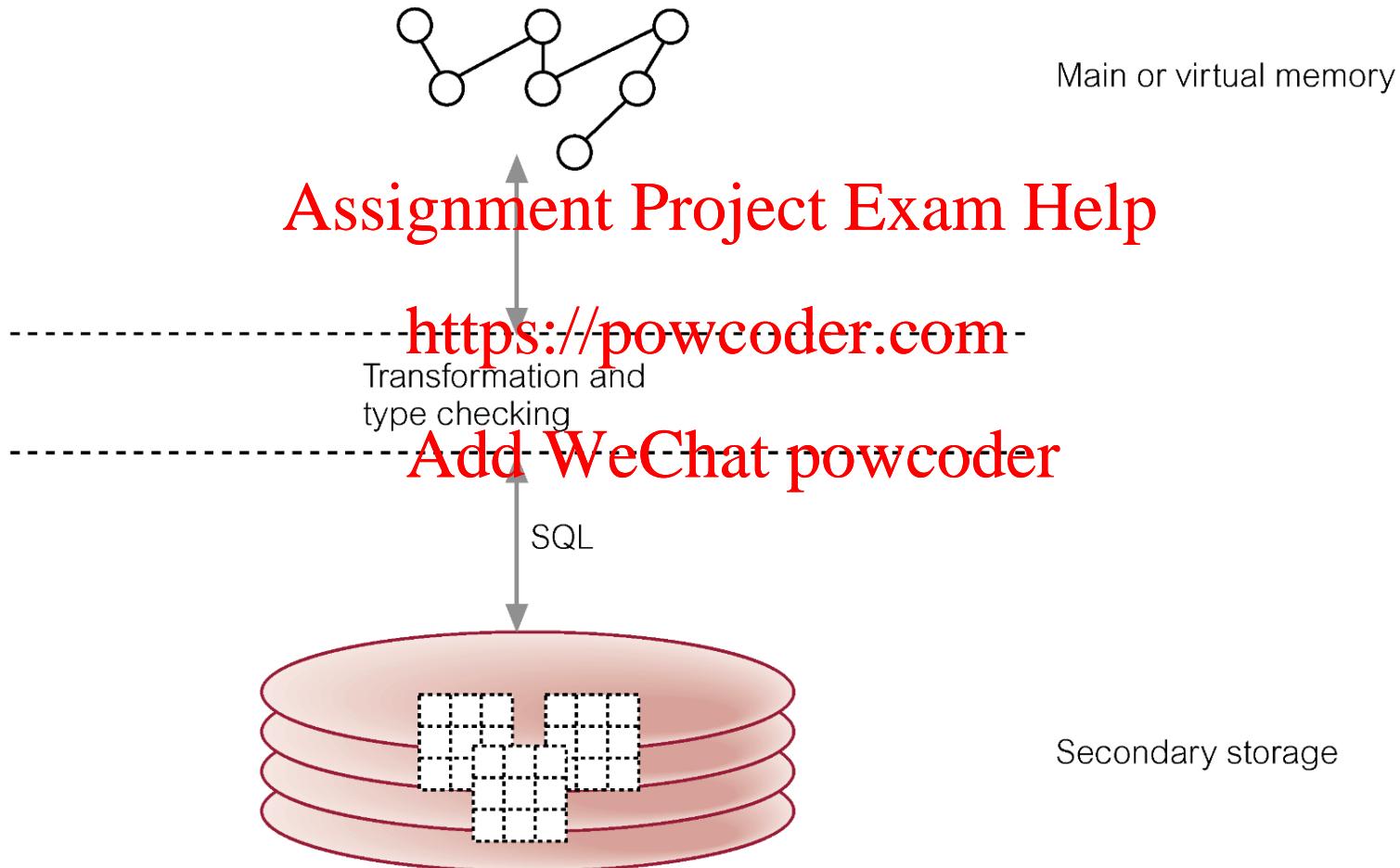
# Single-Level v. Two-Level Storage Model (2 of 3)

- With a traditional DBMS, programmer has to:
  - Decide when to read and update objects.
  - Write code to translate between application's object model and the data model of the DBMS.  
<https://powcoder.com>
  - Perform additional type-checking when object is read back from database, to guarantee object will conform to its original type.

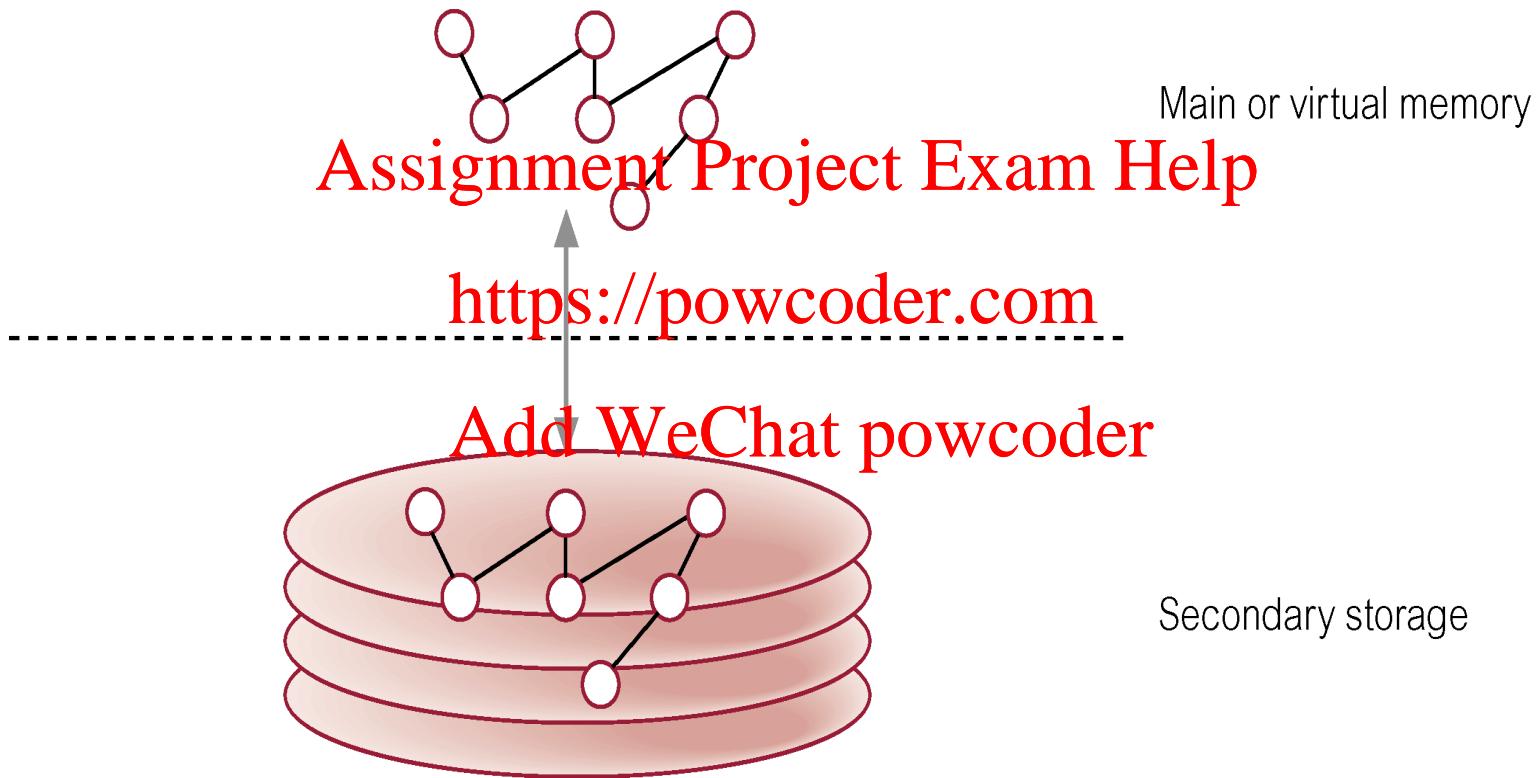
# Single-Level v. Two-Level Storage Model (3 of 3)

- Difficulties occur because conventional DBMSs have two-level storage model: storage model in memory, and database storage model on disk.  
**Assignment Project Exam Help**
- In contrast, OODBMS gives illusion of single-level storage model, with similar representation in both memory and in database stored on disk.  
**Add WeChat powcoder**
  - Requires clever management of representation of objects in memory and on disk (called “pointer swizzling”).

# Two-Level Storage Model for RDBMS



# Single-Level Storage Model for OODBMS



# Pointer Swizzling Techniques (1 of 2)

The action of converting object identifiers (OIDs) to main memory pointers.

- Aim is to optimize access to objects.
- Should be able to locate any referenced objects on secondary storage using their OIDs.  
<https://powcoder.com>  
Add WeChat powcoder
- Once objects have been read into cache, want to record that objects are now in memory to prevent them from being retrieved again.

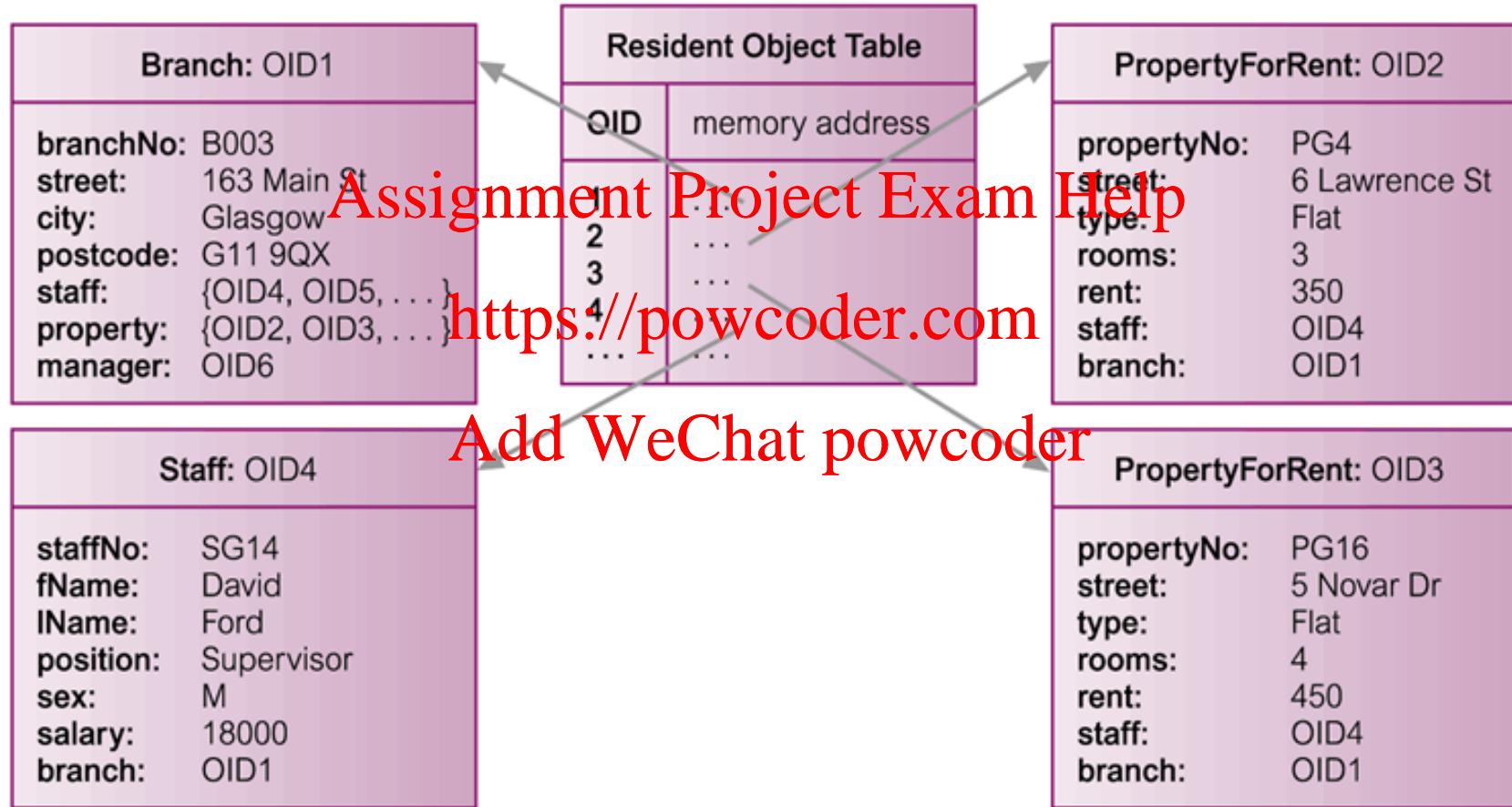
# Pointer Swizzling Techniques (2 of 2)

- Could hold lookup table that maps OIDs to memory pointers (e.g. using hashing).
- Pointer swizzling attempts to provide a more efficient strategy by storing memory pointers in the place of referenced OIDs, and vice versa when the object is written back to disk.

# No Swizzling

- Easiest implementation is not to do any swizzling.
- Objects faulted into memory, and handle passed to application containing object's OID.  
*Assignment Project Exam Help*
- OID is used every time the object is accessed.  
*https://powcoder.com*
- System must maintain some type of lookup table - Resident Object Table (ROT) - so that object's virtual memory pointer can be located and then used to access object.  
*Add WeChat powcoder*
- Inefficient if same objects are accessed repeatedly.
- Acceptable if objects only accessed once.

# Resident Object Table (ROT)



# Object Referencing (1 of 2)

- Need to distinguish between resident and non-resident objects.
- Most techniques variations of edge marking or node marking.  
<https://powcoder.com>
- Edge marking marks every object pointer with a tag bit:  
[Add WeChat powcoder](#)
  - if bit set, reference is to memory pointer;
  - else, still pointing to OID and needs to be swizzled when object it refers to is faulted into.

# Object Referencing (2 of 2)

- Node marking requires that all object references are immediately converted to virtual memory pointers when object is faulted into memory
- First approach is software-based technique but second can be implemented using software or hardware-based techniques.

[Assignment Project Exam Help](#)

<https://powcoder.com>

[Add WeChat powcoder](#)

# Hardware-Based Schemes

- Use virtual memory access protection violations to detect accesses of non-resident objects.
- Use standard virtual memory hardware to trigger transfer of persistent data from disk to memory.  
<https://powcoder.com>
- Once page has been faulted in, objects are accessed via normal virtual memory pointers and no further object residency checking is required.
- Avoids overhead of residency checks incurred by software approaches.

# Pointer Swizzling - Other Issues

- Three other issues that affect swizzling techniques:
  - Copy versus In-Place Swizzling.
  - Eager versus Lazy Swizzling.
  - Direct versus Indirect Swizzling

Add WeChat powcoder

# Copy Versus In-Place Swizzling

- When faulting objects in, data can either be copied into application's local object cache or accessed in-place within object manager's database cache.
- Copy swizzling may be more efficient as, in the worst case, only modified objects have to be swizzled back to their OIDs.
- In-place may have to unswizzle entire page of objects if one object on page is modified.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

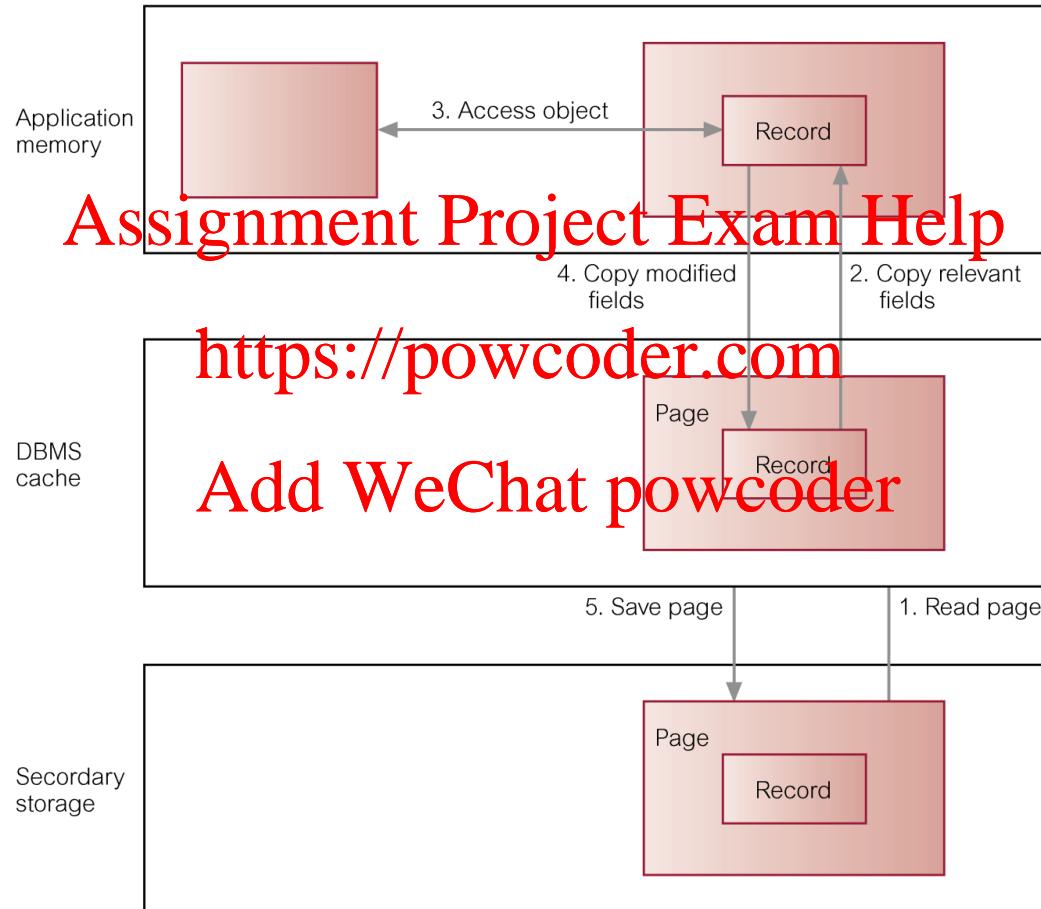
# Eager Versus Lazy Swizzling

- Moss defines eager swizzling as swizzling all OIDs for persistent objects on all data pages used by application, before any object can be accessed.
- More relaxed definition restricts swizzling to all persistent OIDs within object the application wishes to access.
- Lazy swizzling only swizzles pointers as they are accessed or discovered.

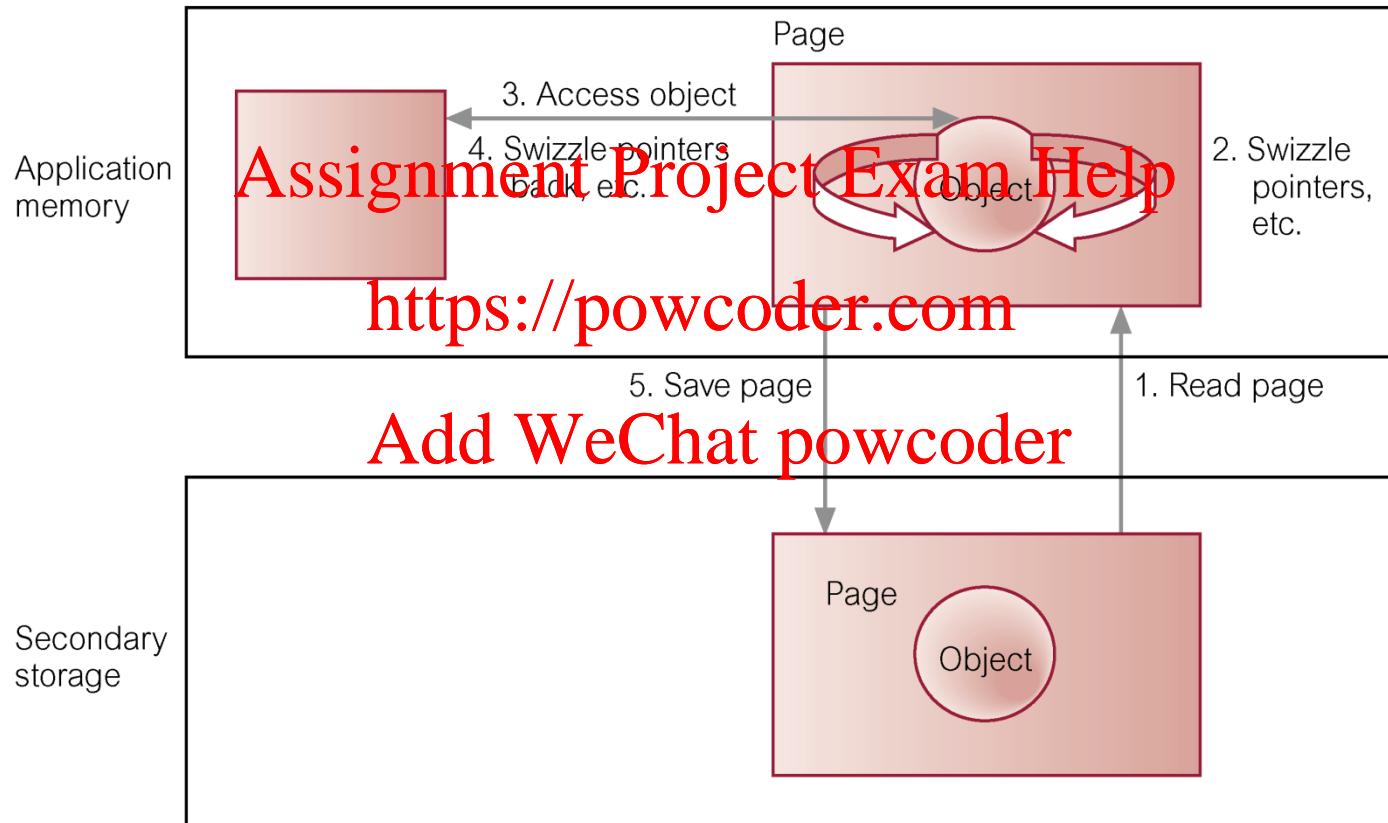
# Direct versus Indirect Swizzling

- Only an issue when swizzled pointer can refer to object that is no longer in virtual memory.
- With direct swizzling, virtual memory pointer of referenced object is placed directly in swizzled pointer.  
<https://powcoder.com>
- With indirect swizzling, virtual memory pointer is placed in an intermediate object, which acts as a placeholder for the actual object.
  - Allows objects to be uncached without requiring swizzled pointers to be unswizzled.

# Accessing an Object with a RDBMS



# Accessing an Object with an ODBMS



# Persistent Schemes

- Consider three persistent schemes:
  - Checkpointing.
  - Serialization.
  - Explicit Paging
- Note, persistence can also be applied to (object) code and to the program execution state.

# Checkpointing

- Copy all or part of program's address space to secondary storage.
- If complete address space saved, program can restart from checkpoint.  
<https://powcoder.com>
- In other cases, only program's heap saved.  
[Add WeChat powcoder](#)
- Two main drawbacks:
  - Can only be used by program that created it.
  - May contain large amount of data that is of no use in subsequent executions.

# Serialization (1 of 2)

- Copy closure of a data structure to disk.
- Write on a data value may involve traversal of graph of objects reachable from the value, and writing of flattened version of structure to disk.  
<https://powcoder.com>
- Reading back flattened data structure produces new copy of original data structure.
- Sometimes called serialization, pickling, or in a distributed computing context, marshaling.

# Serialization (2 of 2)

- Two inherent problems:
  - Does not preserve object identity.
  - Not incremental, so saving small changes to a large data structure is not efficient.

<https://powcoder.com>

Add WeChat powcoder

# Explicit Paging

- Explicitly ‘page’ objects between application heap and persistent store.
- Usually requires conversion of object pointers from disk-based scheme to memory-based scheme.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- Two common methods for creating/updating persistent objects:
  - Reachability-based.
  - Allocation-based.

# Explicit Paging - Reachability-Based Persistence

- Object will persist if it is reachable from a persistent root object.
- Programmer does not need to decide at object creation time whether object should be persistent.  
<https://powcoder.com>
- Object can become persistent by adding it to the reachability tree.  
[Add WeChat powcoder](#)
- Maps well onto language that contains garbage collection mechanism (e.g. Smalltalk or Java).

# Explicit Paging - Allocation-Based Persistence (1 of 2)

- Object only made persistent if it is explicitly declared as such within the application program.
- Can be achieved in several ways.
  - By class. <https://powcoder.com>
  - By explicit call.  
**Add WeChat powcoder**

# Explicit Paging - Allocation-Based Persistence (2 of 2)

- By class
  - Class is statically declared to be persistent and all instances made persistent when they are created.
  - Class may be subclass of system-supplied persistent class.
- By explicit call [Add WeChat powcoder](https://powcoder.com)
  - Object may be specified as persistent when it is created or dynamically at runtime.

# Orthogonal Persistence

- Three fundamental principles:
  - Persistence independence.
  - Data type orthogonality.
  - Transitive persistence (originally referred to as ‘persistence identification’ but ODMG term ‘transitive persistence’  
<https://powcoder.com>  
Added WeChat powcoder

# Persistence Independence

- Persistence of object independent of how program manipulates that object.
- Conversely, code fragment independent of persistence of data it manipulates.  
<https://powcoder.com>
- Should be possible to call function with its parameters sometimes objects with long term persistence and sometimes only transient.
- Programmer does not need to control movement of data between long-term and short-term storage.

# Data Type Orthogonality

- All data objects should be allowed full range of persistence irrespective of their type.
- No special cases where object is not allowed to be long-lived or is not allowed to be transient.  
<https://powcoder.com>
- In some PPLs, persistence is quality attributable to only subset of language data types.  
[Add WeChat powcoder](#)

# Transitive Persistence

- Choice of how to identify and provide persistent objects at language level is independent of the choice of data types in the language.
- Technique that is now widely used for identification is reachability-based.

Add WeChat powcoder

# Orthogonal Persistence - Advantages

- Improved programmer productivity from simpler semantics.
- Improved maintenance.
- Consistent protection mechanisms over whole environment.
- Support for incremental evolution.
- Automatic referential integrity.

Add WeChat powcoder

# Orthogonal Persistence - Disadvantages

- Some runtime expense in a system where every pointer reference might be addressing persistent object.
  - System ~~Assignment Project Exam Help~~ loaded in from disk-resident database.  
<https://powcoder.com>
- Although orthogonal persistence promotes transparency, system with support for sharing among concurrent processes cannot be fully transparent.

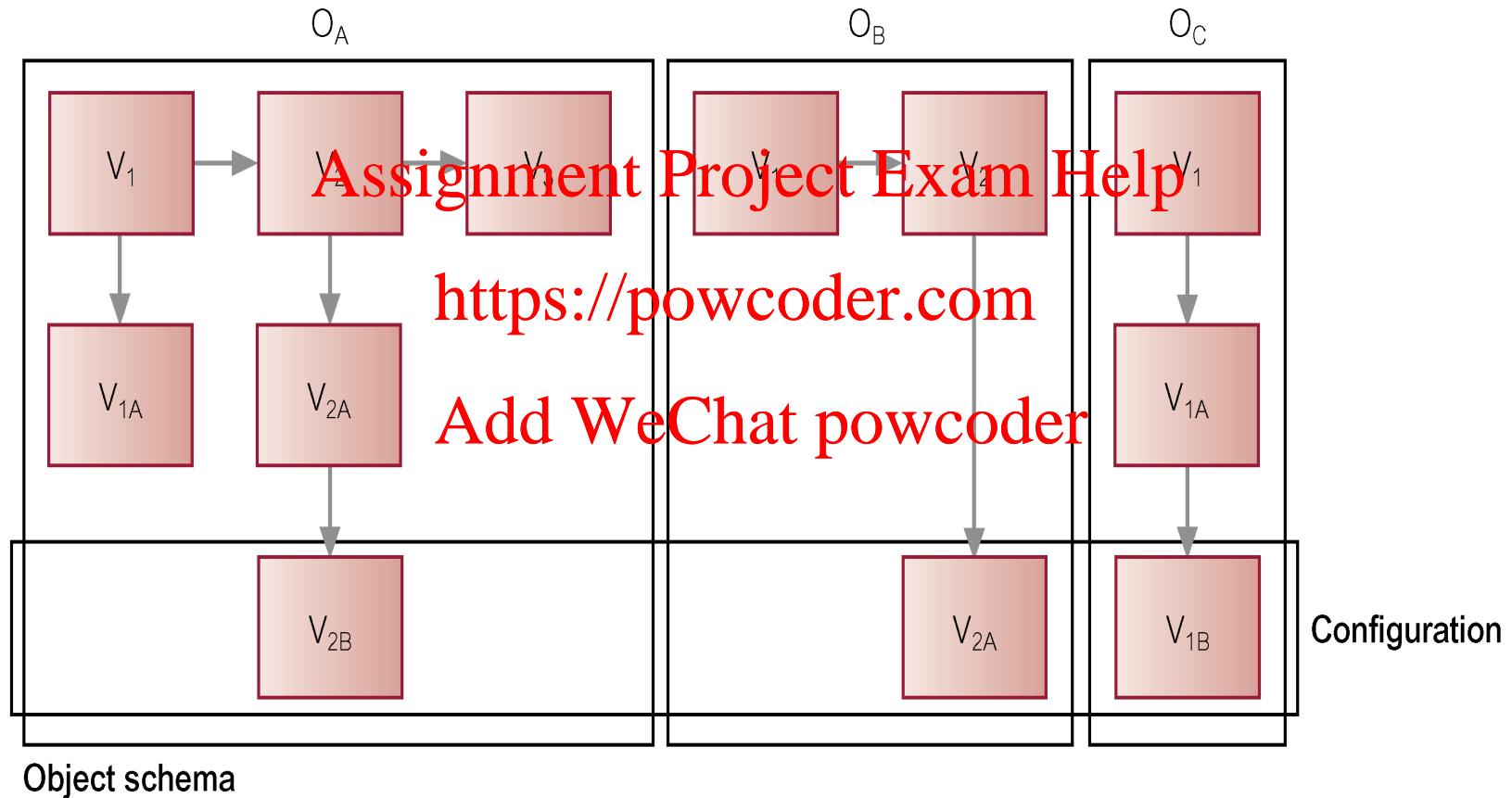
# Versions

Allows changes to properties of objects to be managed so that object references always point to correct object version.

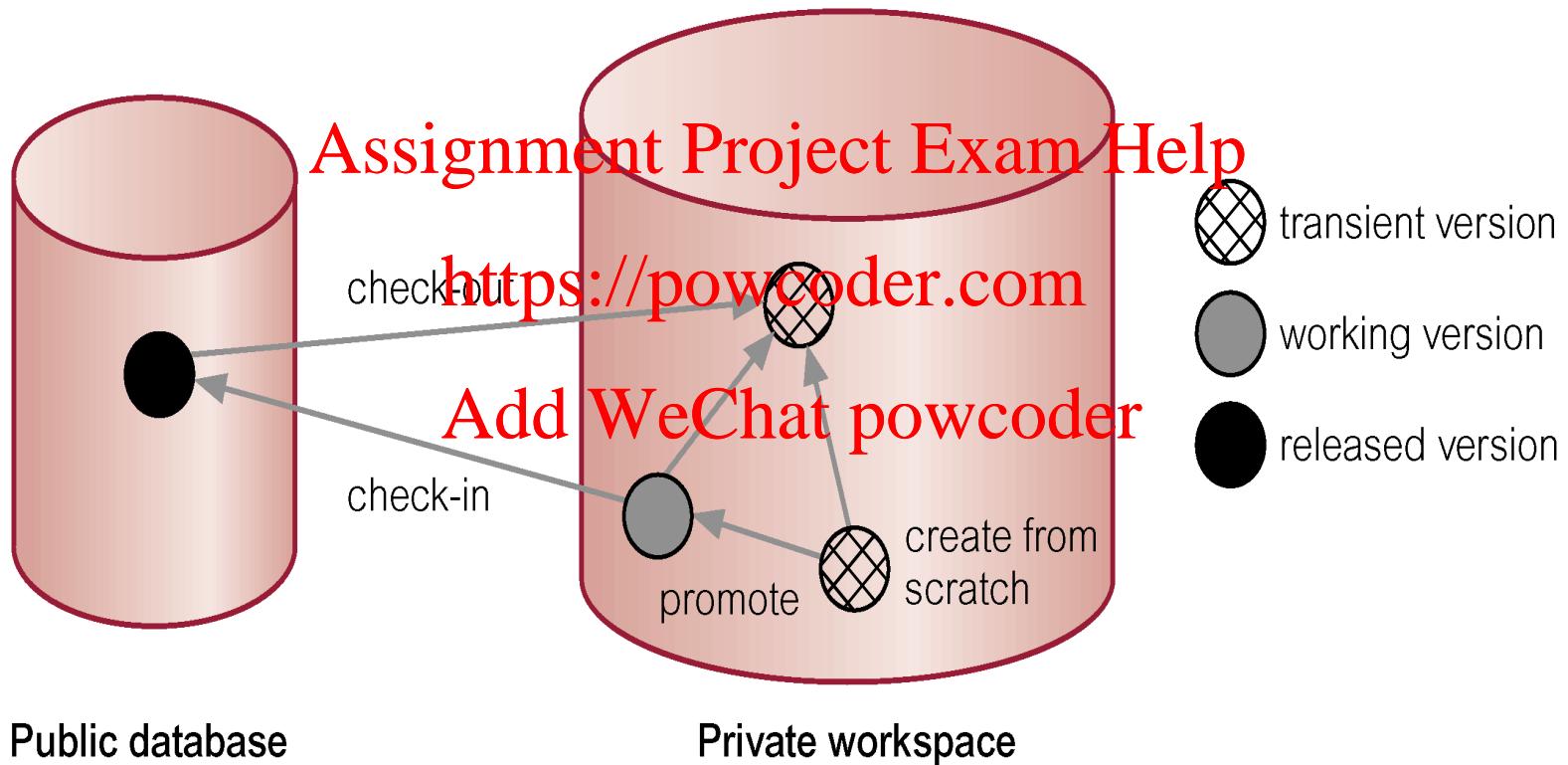
[Assignment](#) [Project](#) [Exam](#) [Help](#)

- Itasca identifies 3 types of versions:  
<https://powcoder.com>
  - Transient Versions.
  - Working Versions.  
[Add WeChat powcoder](#)
  - Released Versions.

# Versions and Configurations (1 of 2)



# Versions and Configurations (2 of 2)



# Schema Evolution (1 of 2)

- Some applications require considerable flexibility in dynamically defining and modifying database schema.
- Typical schema changes:  
**(1) Changes to class definition:**
  - (a) Modifying Attributes.
  - (b) Modifying Methods.

## Schema Evolution (2 of 2)

- (2) Changes to inheritance hierarchy:
- (a) Making a class S superclass of a class C.
  - (b) Removing S from list of superclasses of C.
  - (c) Modifying order of superclasses of C.
- (3) Changes to set of classes, such as creating and deleting classes and modifying class names.
- Changes must not leave schema inconsistent.

# Schema Consistency (1 of 4)

1. Resolution of conflicts caused by multiple inheritance and redefinition of attributes and methods in a subclass.
  - 1.1 Rule of precedence of subclasses over superclasses.
  - 1.2 Rule of precedence between superclasses of a different origin.
  - 1.3 Rule of precedence between superclasses of the same origin.

Add WeChat powcoder

# Schema Consistency (2 of 4)

2. Propagation of modifications to subclasses.

2.1 Rule for propagation of modifications.

Assignment Project Exam Help

2.2 Rule for propagation of modifications in the event of conflicts. <https://powcoder.com>

2.3 Rule for modification of domains.

Add WeChat powcoder

# Schema Consistency (3 of 4)

3. Aggregation and deletion of inheritance relationships between classes and creation and removal of classes.

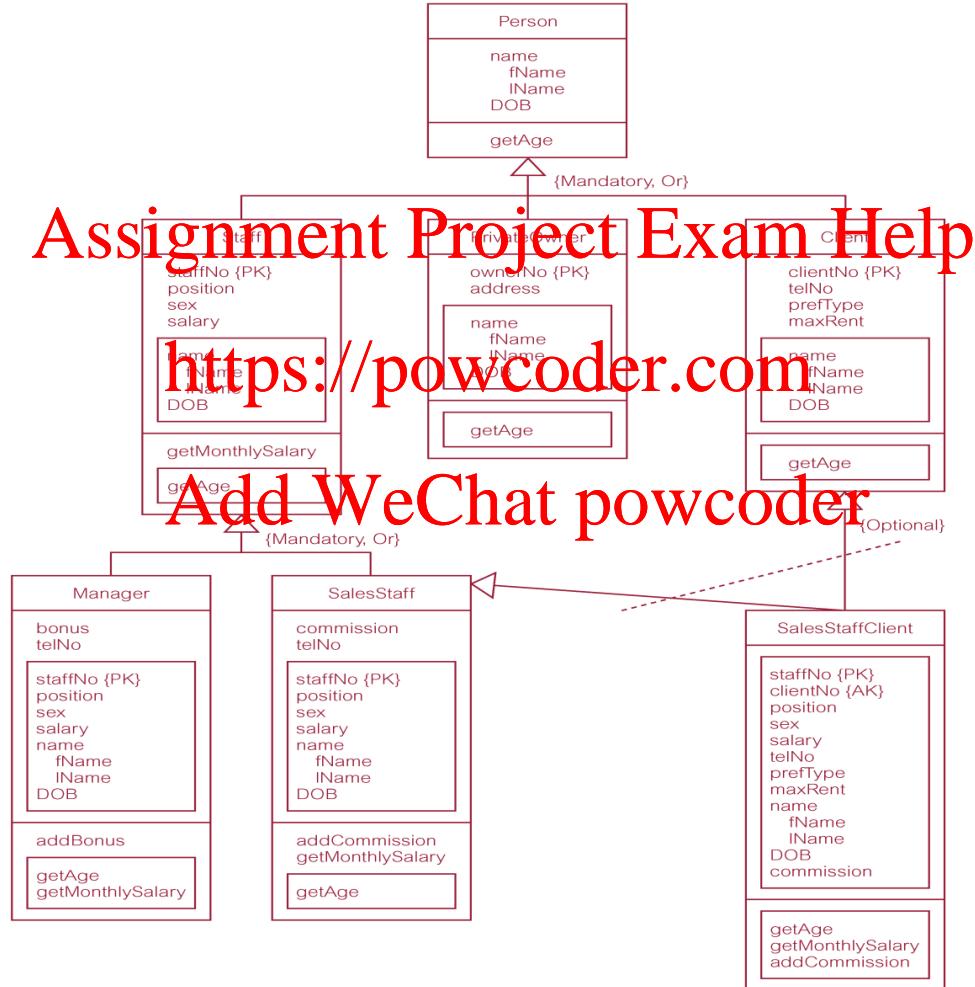
3.1 Rule for inserting superclasses.

3.2 Rule for removing ~~superclasses.~~ <https://powcoder.com>

3.3 Rule for inserting ~~Add WeChat~~ [powcoder](https://powcoder.com) into a schema.

3.4 Rule for removing a class from a schema.

# Schema Consistency (4 of 4)



# Client-Server Architecture (1 of 2)

- Three basic architectures:
  - Object Server.
  - Page Server
  - Database Server

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Object Server

- Distribute processing between the two components.
- Typically, client is responsible for transaction management and interfacing to programming language.
- Server responsible for other DBMS functions.
- Best for cooperative, object-to-object processing in an open, distributed environment.

# Page and Database Server

## Page Server

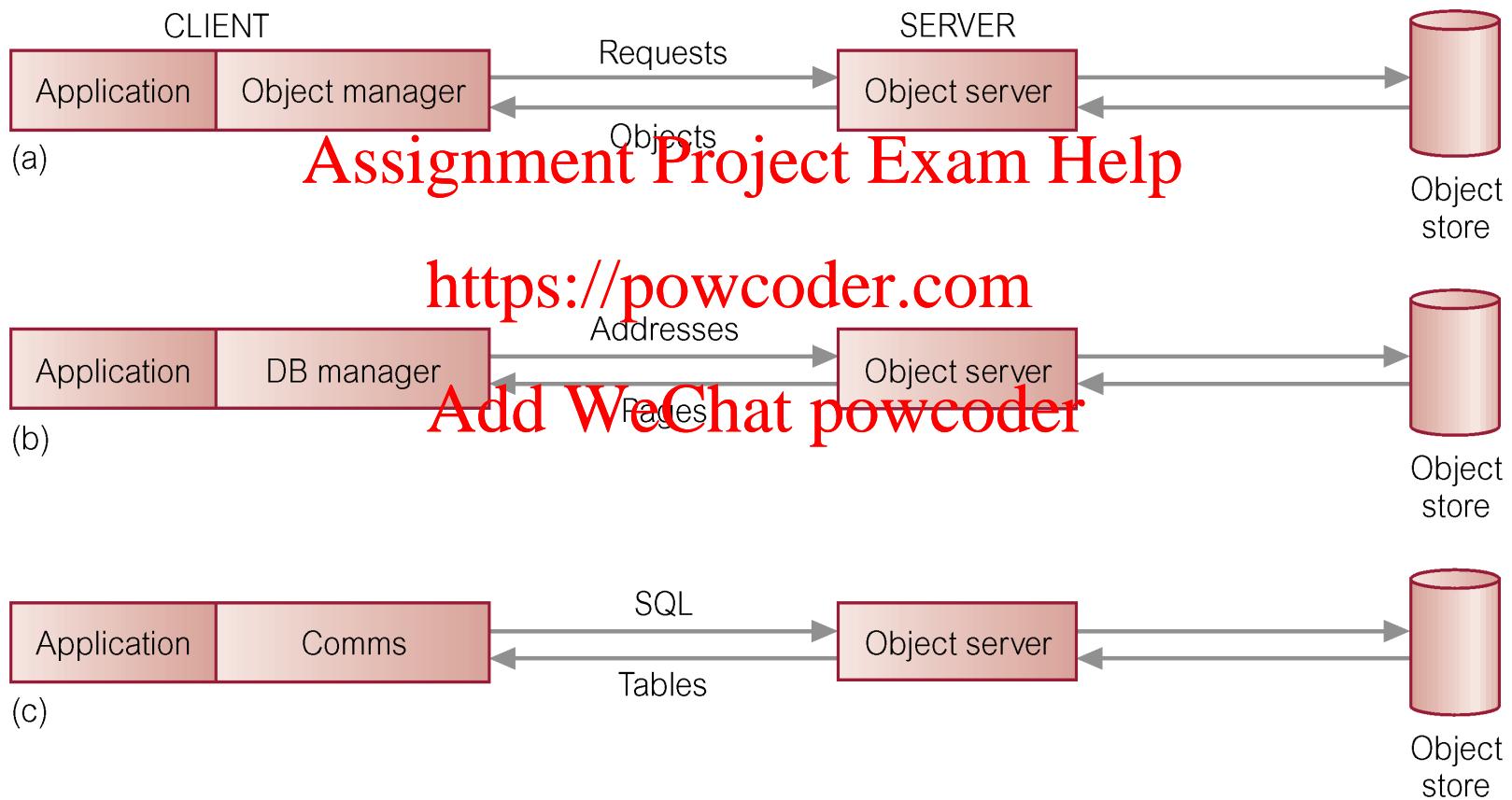
- Most database processing is performed by client.  
**Assignment Project Exam Help**
- Server responsible for secondary storage and providing pages at client's request.  
**<https://powcoder.com>**

## Database Server

Add WeChat powcoder

- Most database processing performed by server.
- Client simply passes requests to server, receives results and passes them to application.
- Approach taken by many RDBMSs.

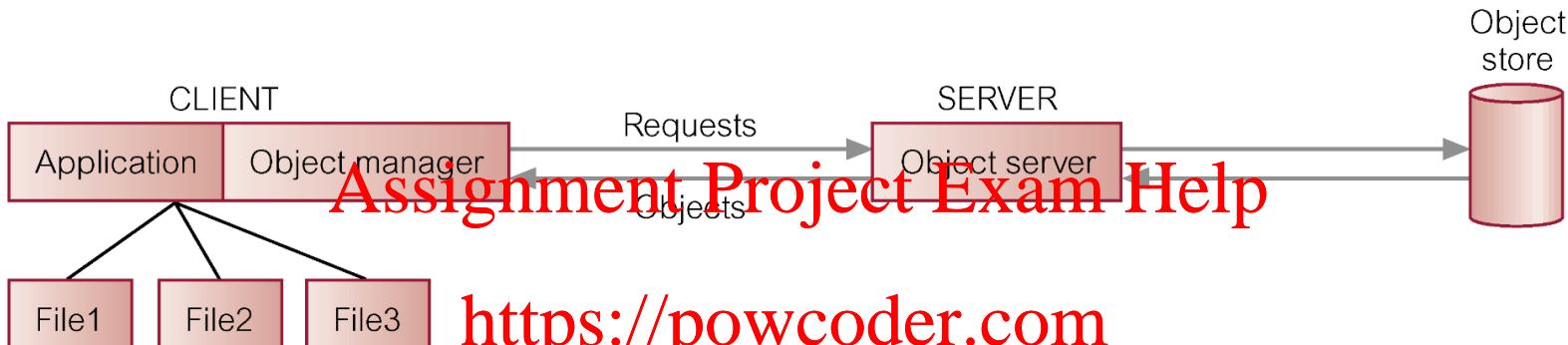
# Client-Server Architecture (2 of 2)



# Architecture - Storing and Executing Methods (1 of 2)

- Two approaches:
  - Store methods in external files.
  - Store methods in database.
- Benefits of latter
  - Eliminates redundant code.
  - Simplifies modifications.
  - Methods are more secure.
  - Methods can be shared concurrently.
  - Improved integrity.
- Obviously, more difficult to implement.

# Architecture - Storing and Executing Methods (2 of 2)

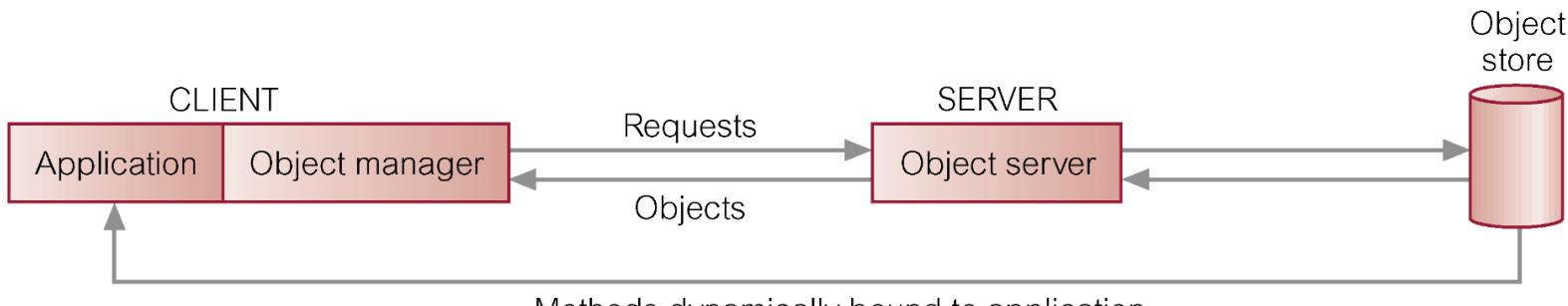


Methods in files (bound at compile and link time)

(a)

<https://powcoder.com>

Add WeChat powcoder



(b)

Methods dynamically bound to application

# Benchmarking - Wisconsin Benchmark (1 of 2)

- Developed to allow comparison of particular DBMS features.
- Consists of set of tests as a single user covering:
  - updates/deletes involving key and non-key attributes;
  - projections involving different degrees of duplication in the attributes and selections with different selectivities on indexed, non-index, and clustered attributes;
  - joins with different selectivities;
  - aggregate functions.

# Benchmarking - Wisconsin Benchmark (2 of 2)

- Original benchmark had 3 relations: one called Onektup with 1000 tuples, and two others called Tenktup1/Tenktup2 with 10000 tuples
- Generally useful although does not cater for highly skewed attribute distributions and join queries used are relatively simplistic.
- Consortium of manufacturers formed Transaction Processing Council (TPC) in 1988 to create series of transaction-based test suites to measure database/TP environments.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# TPC Benchmarks

- TPC-A and TPC-B for OLTP (now obsolete).
- TPC-C replaced TPC-A/B and based on order entry application.
- TPC-H for ad hoc, decision support environments.
- TPC-R for business reporting within decision support environments.
- TPC-W, a transactional Web benchmark for eCommerce.

# Object Operations Version 1 (OO1) Benchmark

- Intended as generic measure of OODBMS performance. Designed to reproduce operations common in advanced engineering applications such as finding all parts connected to a random part, all parts connected to one of those parts, and so on, to a depth of seven levels.
- About 1990, benchmark was run on GemStone, Ontos, ObjectStore, Objectivity/DB, and Versant, and INGRES and Sybase. Results showed an average 30-fold performance improvement for OODBMSs over RDBMSs.

# OO7 Benchmark

- More comprehensive set of tests and a more complex database based on parts hierarchy.
- Designed for detailed comparisons of OODBMS products. [Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- Simulates CAD/CAM environment and tests system performance in area of object-to-object navigation over cached data, disk-resident data, and both sparse and dense traversals.
- Also tests indexed and nonindexed updates of objects, repeated updates, and the creation and deletion of objects.

# Advantages of OODBMSs

- Enriched Modeling Capabilities.
- Extensibility.  
[Assignment Project Exam Help](#)  
<https://powcoder.com>
- Removal of Impedance Mismatch.
- More Expressive Query Language.
- Support for Schema Evolution.  
[Add WeChat powcoder](#)
- Support for Long Duration Transactions.
- Applicability to Advanced Database Applications.
- Improved Performance.

# Disadvantages of OOBMSs

- Lack of Universal Data Model.
- Lack of Experience.  
[Assignment Project Exam Help](#)
- Lack of Standards.  
<https://powcoder.com>
- Query Optimization compromises Encapsulation.
- Object Level Locking may impact Performance.  
[Add WeChat powcoder](#)
- Complexity.
- Lack of Support for Views.
- Lack of Support for Security.

# Comparison of ORDBMS and OODBMS – Data Modeling

Feature	ORDBMS	OODBMS
Object identity (OID)	Supported through REF type	Supported
Encapsulation	Supported through UDTs	Supported but broken for queries
Inheritance	Supported (separate hierarchies for UDTs and tables)	Supported
Polymorphism	Supported (UDF invocation based on the generic function)	Supported as in an object oriented programming model language
Complex objects	Supported through UDTs	Supported
Relationships	Strong support with user-defined referential integrity constraints	Supported (for example, using class libraries)

# Comparison of ORDBMS and OODBMS – Data Access

Feature	ORDBMS	OODBMS
Creating and accessing persistent data	Supported but not transparent	Supported but degree of transparency differs between products
Ad hoc query facility	Strong support <a href="https://powcoder.com">https://powcoder.com</a>	Supported through ODMG 3.0
Navigation	Supported by REF type Add WeChat powcoder	Strong support
Integrity constraints	Strong support	No support
Object server/page server	Object server	Either
Schema evolution	Limited support	Supported but degree of support differs between products

# Comparison of ORDBMS and OODBMS – Data Sharing

Feature	ORDBMS	OODBMS
ACID transactions	Strong support	Supported
Recovery	Strong support	Supported but degree of support differs between products
Advanced transaction models	No support	Supported but degree of support differs between products
Security, integrity, and views	Strong support	Limited support

# OO Database Design – OODB vs. CDM

OODM	CDM	Difference
Object	Entity	Object includes behavior
Attribute	Attribute	None
Association	Relationship	Associations are the same but inheritance in OODM includes both state and behavior
Message		No corresponding concept in CDM  <b>Add WeChat powcoder</b> <a href="https://powcoder.com">https://powcoder.com</a>
Class	Entity type/Supertype	None
Instance	Entity	None
Encapsulation		No corresponding concept in CDM

# Relationships

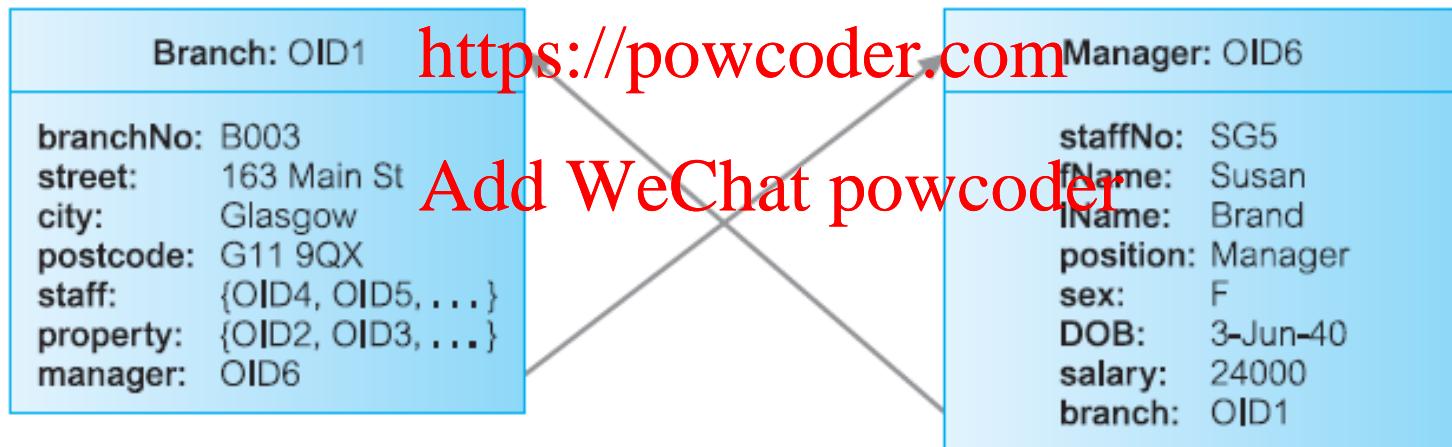
- Relationships represented using reference attributes, typically implemented using OIDs.
- Consider how to represent following binary relationships according to their cardinality:  
<https://powcoder.com>
  - 1 : 1
  - 1 : \*
  - \* : \*

Add WeChat powcoder

# 1 : 1 Relationship Between Objects A and B

- Add reference attribute to A and, to maintain referential integrity, reference attribute to B.

Assignment Project Exam Help



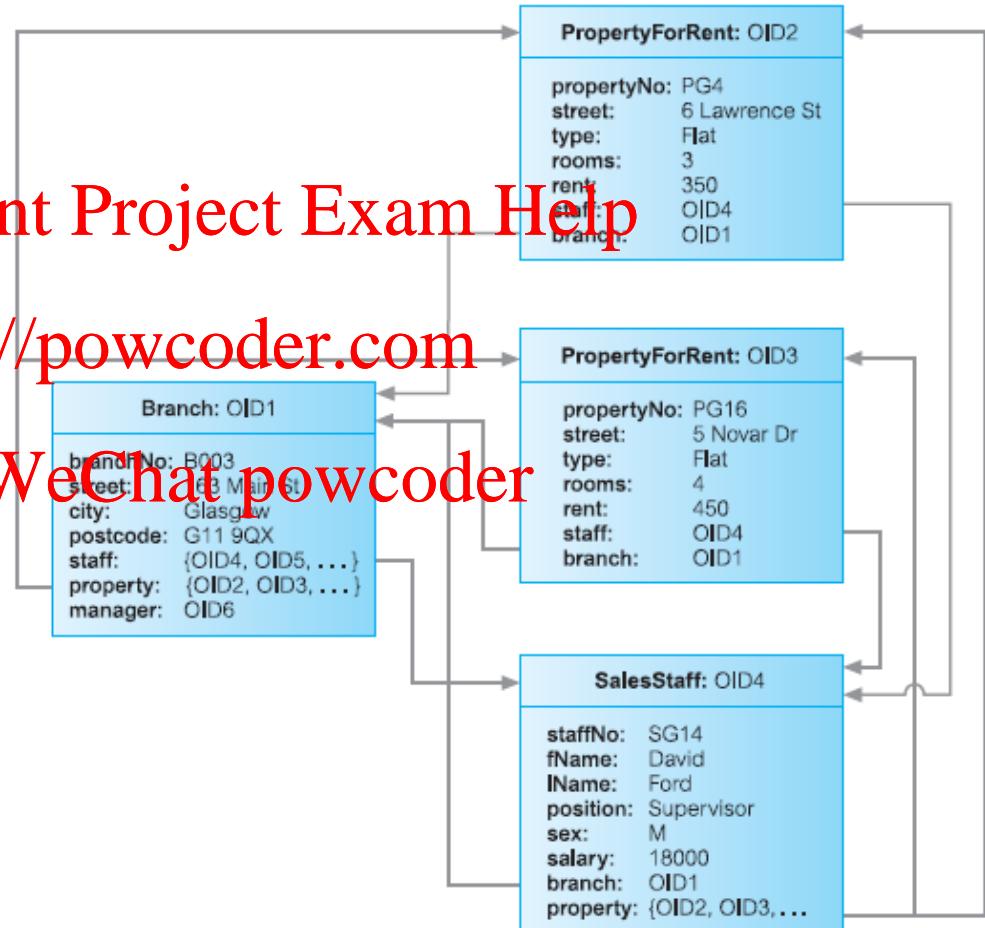
# 1 : \* Relationship Between Objects A and B

- Add reference attribute to B and attribute containing set of references to A.

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

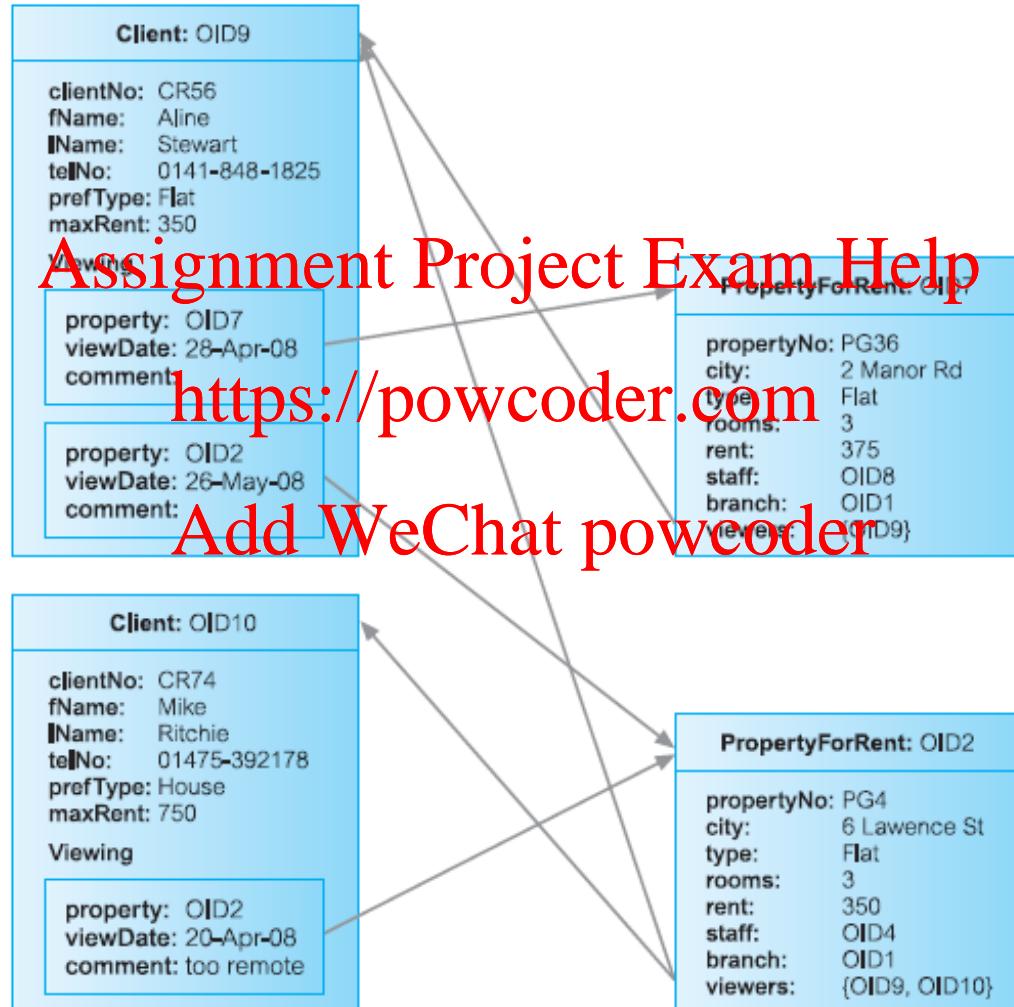


# \* : \* Relationship Between Objects A and B

- Add attribute containing set of references to each object.
- For relational database design, would decompose \* : N  
**Assignment Project Exam Help**  
into two 1 : \* relationships linked by intermediate entity.  
**https://powcoder.com**  
Can also represent this model in an ODBMS.

Add WeChat powcoder

# \* : \* Relationships

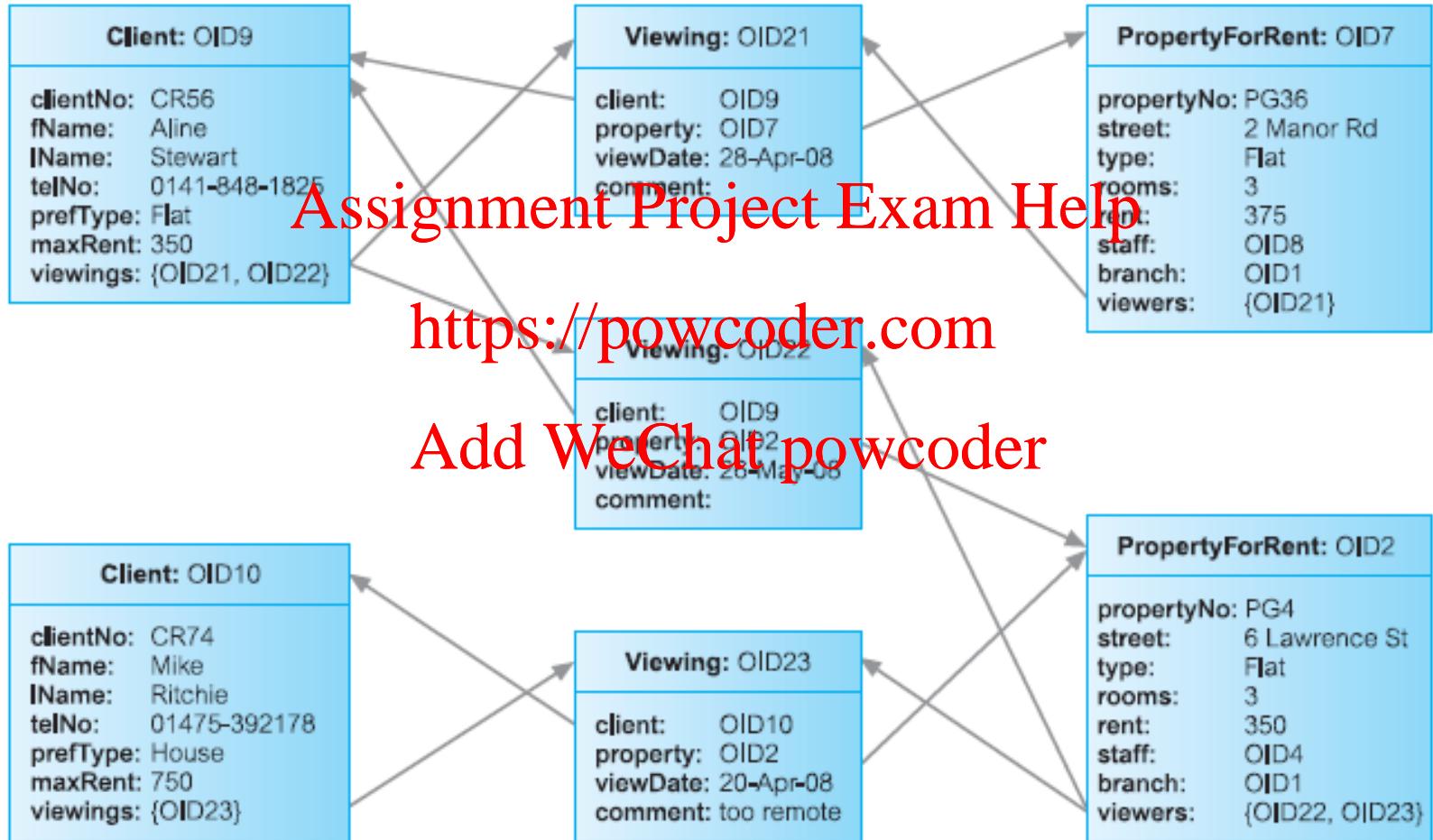


Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Alternative Design for \* : \* Relationships



# Referential Integrity (1 of 2)

Several techniques to handle referential integrity:

- Do not allow user to explicitly delete objects.
  - System is responsible for “garbage collection”.
- Allow user to delete objects when they are no longer required.
  - System may detect invalid references automatically and set reference to NULL or disallow the deletion.

# Referential Integrity (2 of 2)

- Allow user to modify and delete objects and relationships when they are no longer required.
  - System automatically maintains the integrity of objects.
  - Inverse attributes can be used to maintain referential integrity. <https://powcoder.com> Add WeChat powcoder

# Behavioral Design

- EER approach must be supported with technique that identifies behavior of each class.
- Involves identifying.
  - public methods: visible to all users  
<https://powcoder.com>
  - private methods: internal to class.  
Add WeChat powcoder
- Three types of methods:
  - constructors and destructors
  - access
  - transform.

# Behavioral Design - Methods

- Constructor - creates new instance of class.
- Destructor - deletes class instance no longer required.
- Access - returns value of one or more attributes (Get).
- Transform - changes state of class instance (Put).

Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# Identifying Methods

- Several methodologies for identifying methods, typically combine following approaches:
  - Identify ~~Assignment Project Exam Help~~ that may be usefully provided for each class.
  - Decompose application in top-down fashion and determine methods required to provide required functionality.

# UML (1 of 2)

- Represents unification and evolution of several OOAD methods, particularly:
  - Booch [Assignment Project Exam Help](#)
  - Object Modeling Technique (OMT),  
<https://powcoder.com>
  - Object-Oriented Software Engineering (OOSE).
- Adopted as a standard by OMG and accepted by software community as primary notation for modeling objects and components.

## UML (2 of 2)

- Defined as “**a standard language for specifying, constructing, visualizing, and documenting the artifacts of a software system**”  
**Assignment Project Exam Help**
- The UML does not prescribe any particular methodology, but instead is flexible and customizable to fit any approach and can be used in conjunction with a wide range of software lifecycles and development processes.

# UML – Design Goals

- Provide ready-to-use, expressive visual modeling language so users can develop and exchange meaningful models.
- Provide extensibility and specialization mechanisms to extend core concepts.
- Be independent of particular programming languages and development processes.
- Provide a formal basis for understanding the modeling language.
- Encourage growth of object-oriented tools market.
- Support higher-level development concepts such as collaborations, frameworks, patterns, and components.
- Integrate best practices.

**Assignment Project Exam Help**

<https://powcoder.com>

**Add WeChat powcoder**

# UML – Diagrams (1 of 2)

- Structural:
  - class diagrams
  - object diagrams
  - component diagrams
  - deployment diagrams.
- Behavioral:
  - use case diagrams
  - sequence diagrams
  - collaboration diagrams
  - statechart diagrams
  - activity diagrams.

# UML – Diagrams (2 of 2)

- Model instances of classes and used to describe system at a particular point in time.
- Can be used to validate class diagram with “real world” data and record test cases.

Assignment Project Exam Help  
<https://powcoder.com>  
Add WeChat powcoder

SG37: Staff

staffNo = SG37  
fName = Ann  
lName = Beech  
position = Assistant  
sex = F  
DOB = 10-Nov-60  
salary = 12000  
branchNo = B003

Manages

PG36: PropertyForRent

propertyNo = PG36  
street = 2 Manor Rd  
city = Glasgow  
postcode = G32 4QX  
type = Flat  
rooms = 3  
rent = 375  
ownerNo = C093  
staffNo = SG37  
branchNo = B003

PG21: PropertyForRent

propertyNo = PG21  
street = 18 Dale Rd  
city = Glasgow  
postcode = G12  
type = House  
rooms = 5  
rent = 600  
ownerNo = C087  
staffNo = SG37  
branchNo = B003

# UML – Component Diagrams

- Describe organization and dependencies among physical software components, such as source code, run-time (binary) code, and executables

Assignment Project Exam Help

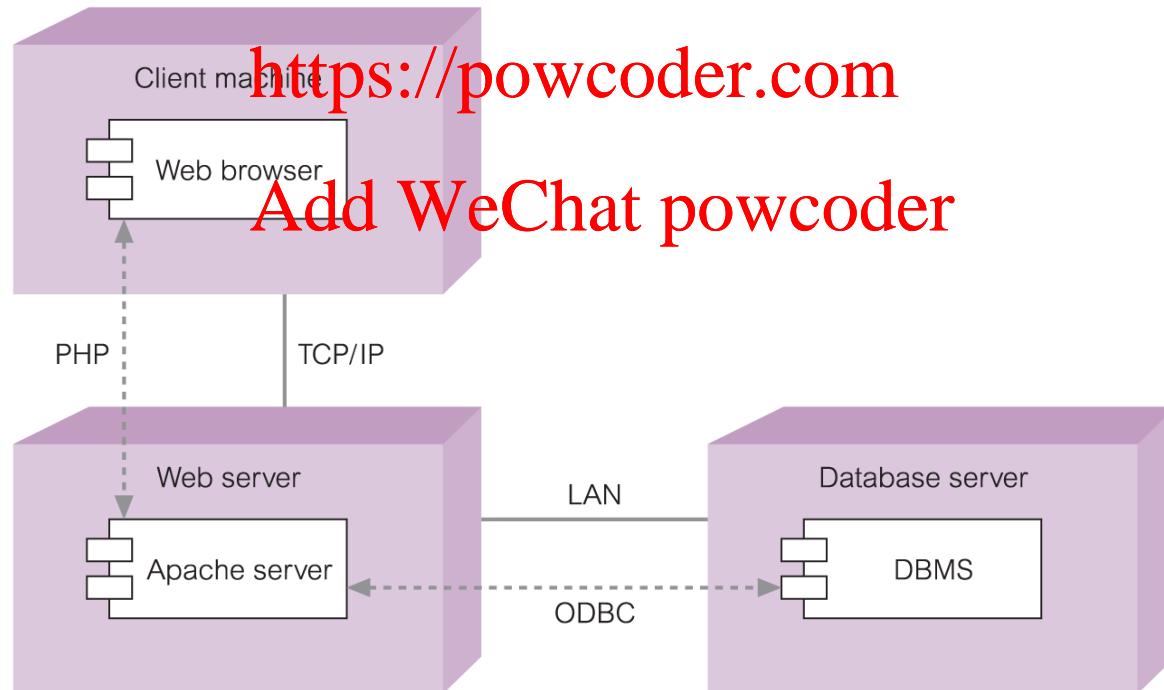
<https://powcoder.com>

Add WeChat powcoder

# UML – Deployment Diagrams

- Depict configuration of run-time system, showing hardware nodes, components that run on these nodes, and connections between nodes

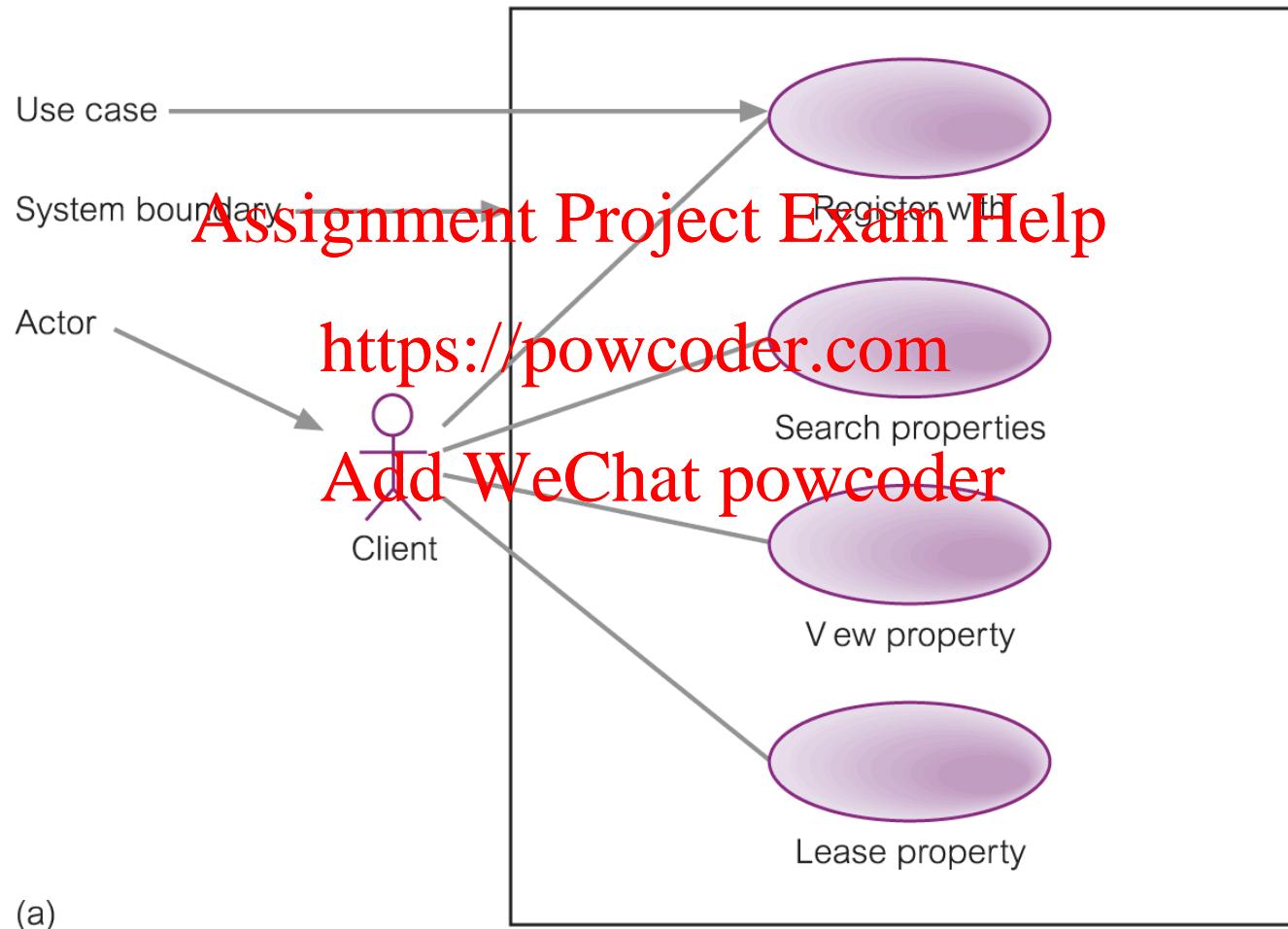
Assignment Project Exam Help



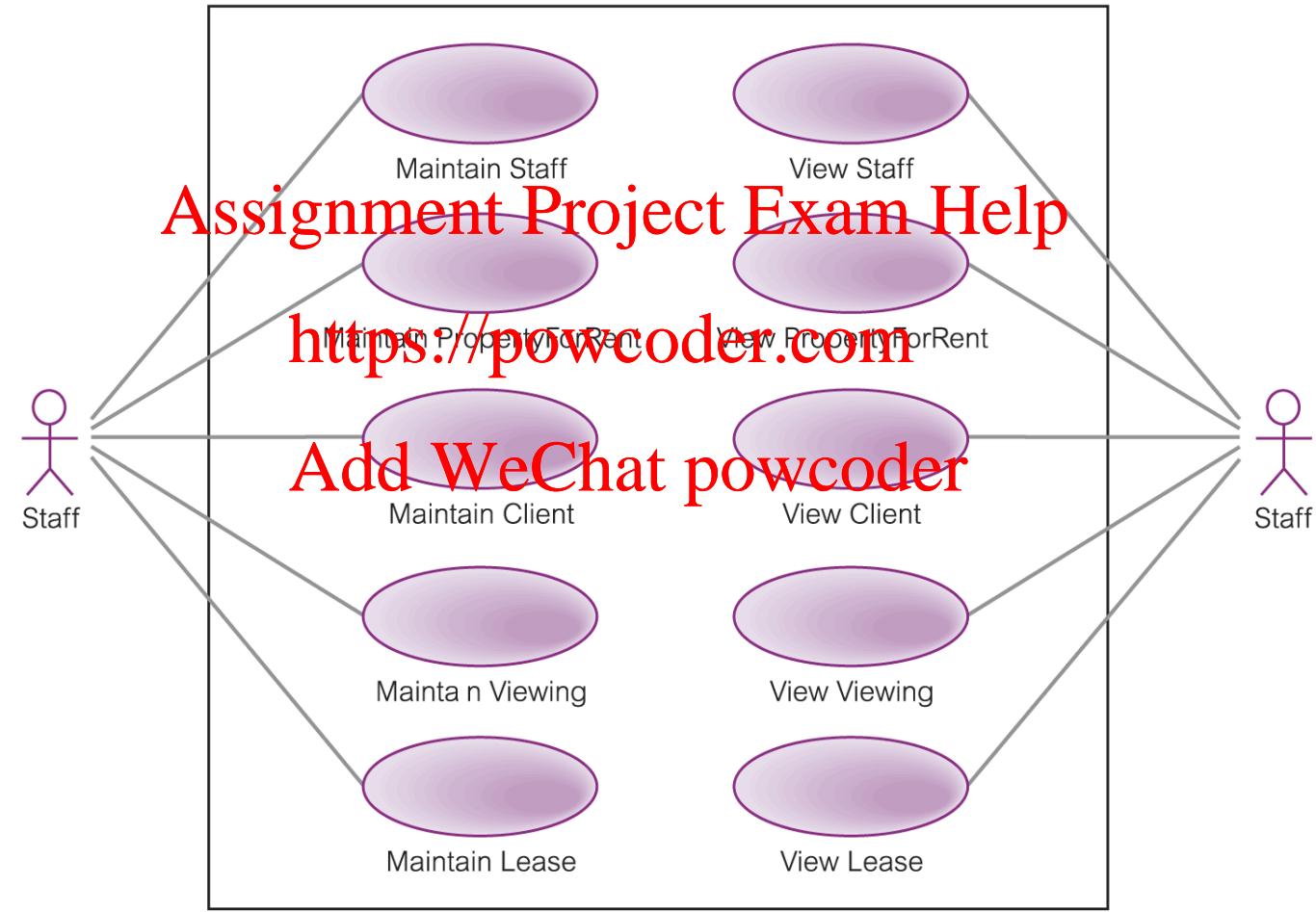
# UML – Use Case Diagrams

- Model functionality provided by system (**use cases**), users who interact with system (**actors**), and association between users and the functionality.
- Used in requirements collection and analysis phase to represent high-level requirements of system.
- More specifically, specifies a sequence of actions, including variants, that system can perform and that yields an observable result of value to a particular actor.

# UML – Use Case Diagrams (1 of 2)



# UML – Use Case Diagrams (2 of 2)



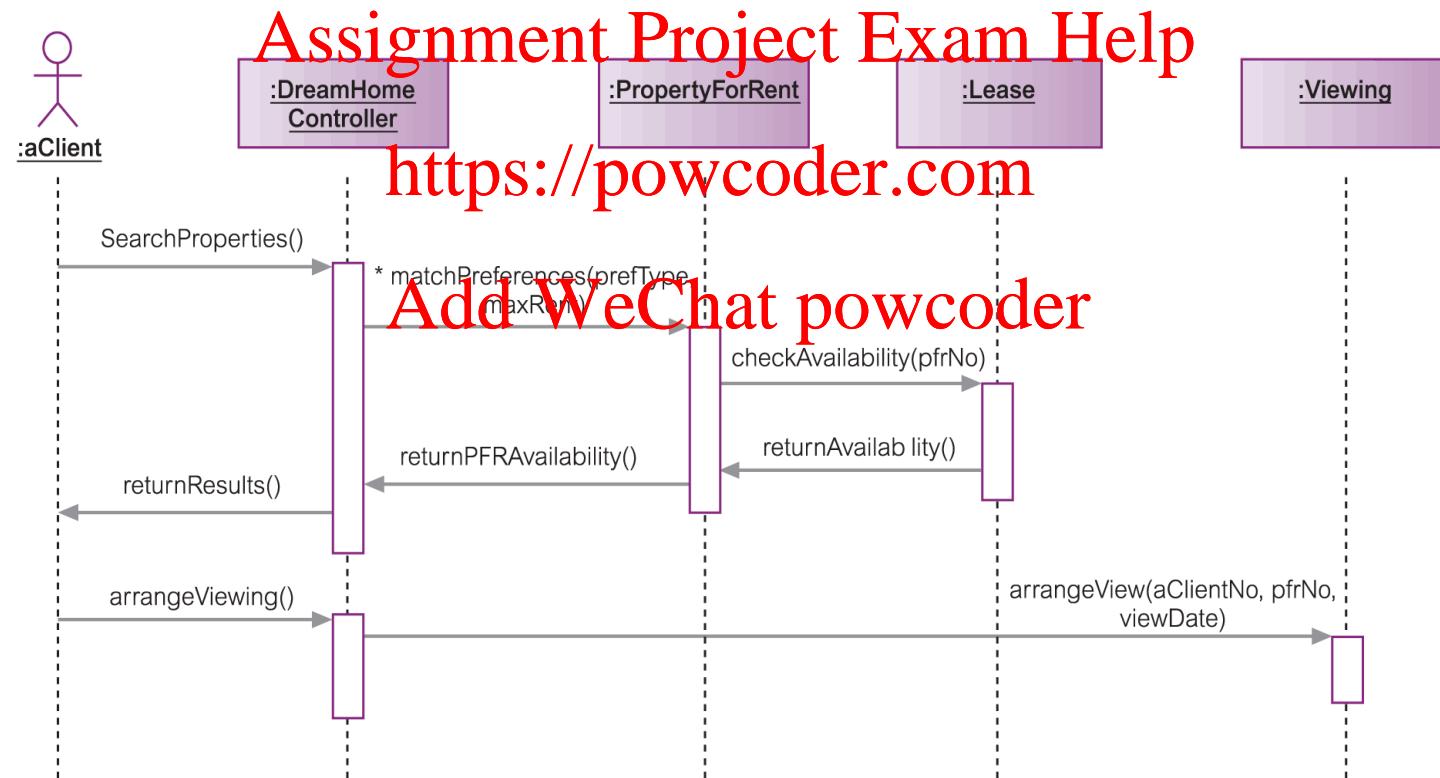
# UML – Sequence Diagrams (1 of 2)

- Model interactions between objects over time, capturing behavior of an individual use case.
- Show the objects and the messages that are passed between these objects in the use case.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)

Add WeChat powcoder

# UML – Sequence Diagrams (2 of 2)

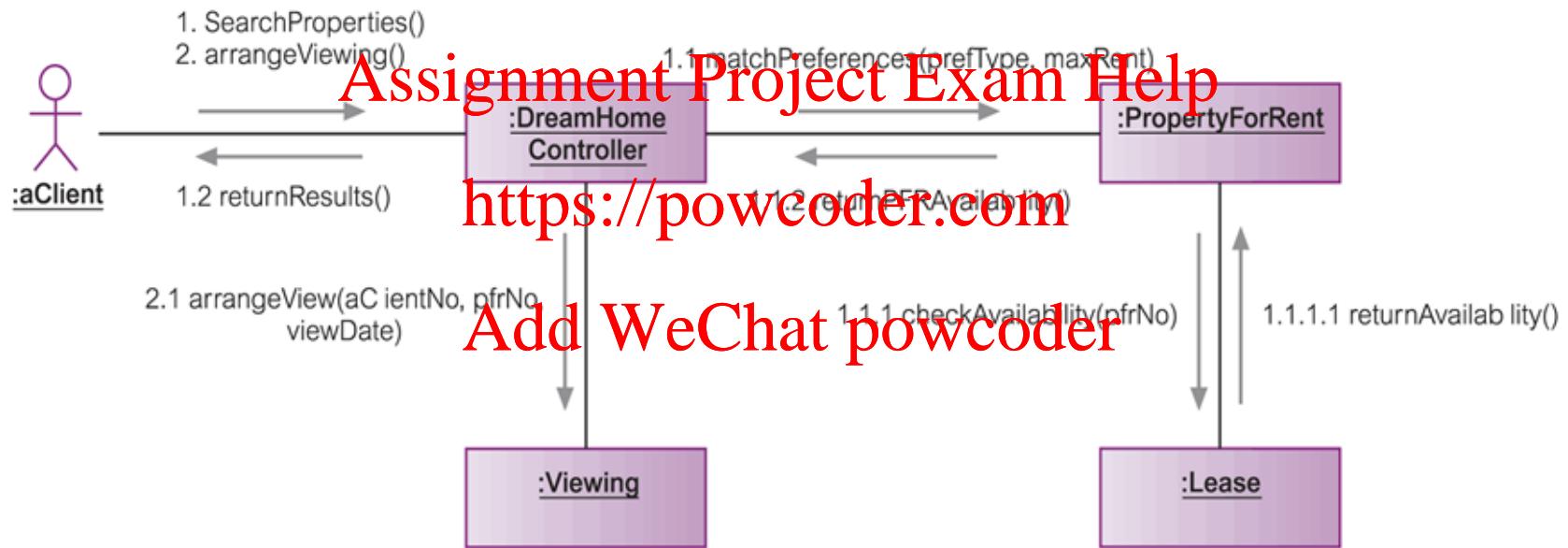
**Figure 25.18** Sequence diagram for search properties use case.



# UML – Collaboration Diagrams (1 of 2)

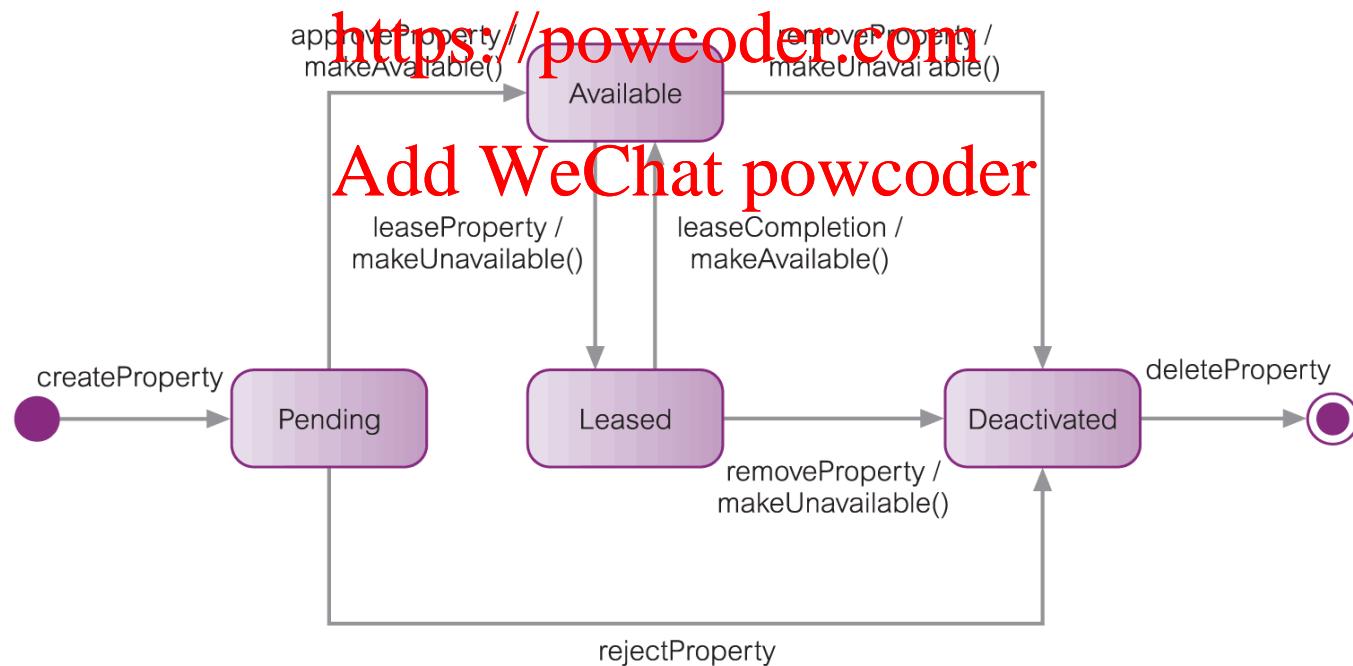
- Show interactions between objects as a series of sequenced messages.
- Cross between an object diagram and a sequence diagram.  
[Assignment Project Exam Help  
https://powcoder.com](https://powcoder.com)
- Unlike sequence diagram, which has column/row format, collaboration diagram uses free-form arrangement, which makes it easier to see all interactions involving a particular object.

# UML – Collaboration Diagrams (2 of 2)



# UML – Statechart Diagrams

- Show how objects can change in response to external events.
- Usually model transitions of a specific object.  
*Assignment Project Exam Help*

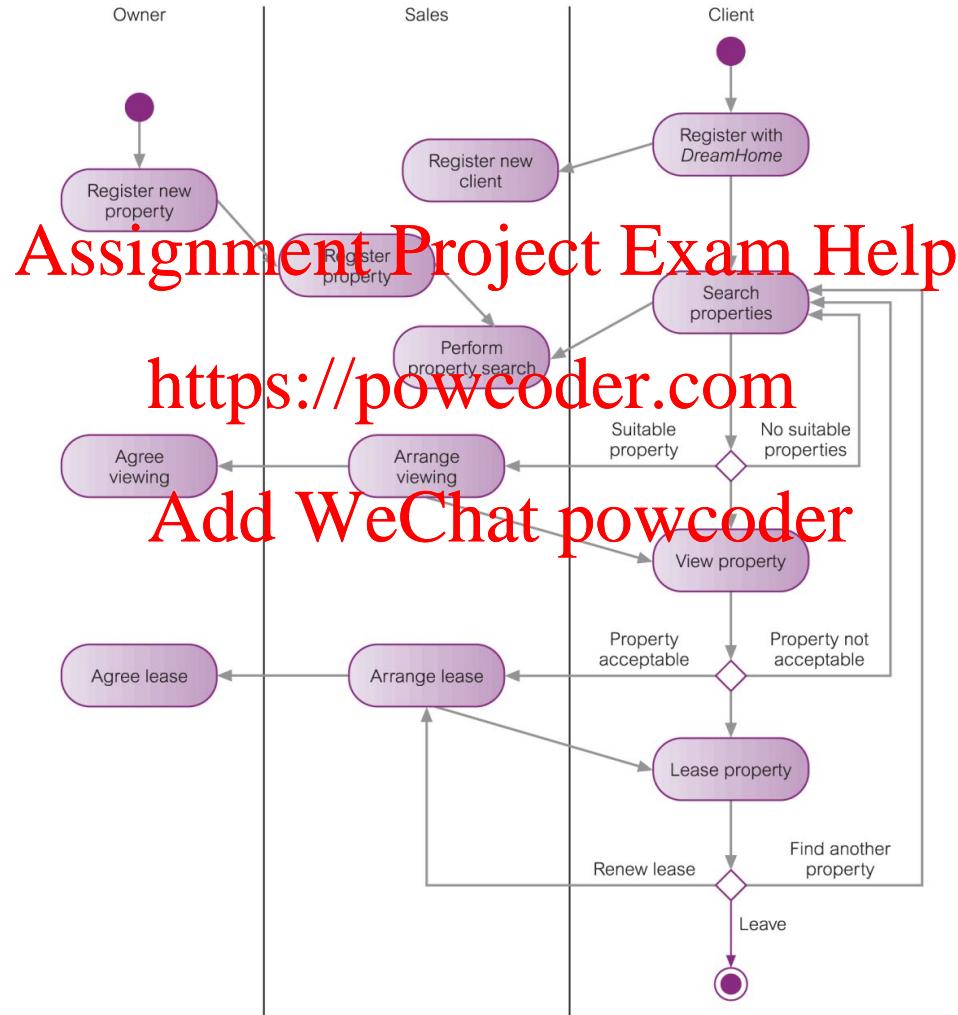


# UML – Activity Diagrams (1 of 2)

- Model flow of control from one activity to another.
- Typically represent invocation of an operation, a step in a business process, or an entire business process.
- Consist of activity states and transitions between them.

Add WeChat powcoder

# UML – Activity Diagrams (2 of 2)



Assignment Project Exam Help

<https://powcoder.com>

Add WeChat powcoder

# UML – Usage in Database Design

## Methodology (1 of 2)

- Produce use case diagrams from requirements specification or while producing requirements specification to depict main functions required of system.  
**Assignment Project Exam Help**  
Can be augmented with use case descriptions.  
<https://powcoder.com>
- Produce first cut class diagram (ER model).
- Produce a sequence diagram for each use case or group of related use cases.
- May be useful to add a **control class** to class diagram to represent interface between the actors and the system.

# UML – Usage in Database Design

## Methodology (2 of 2)

- Update class diagram to show required methods in each class.
- Create state diagram for each class to show how class changes state in response to messages. Messages are identified from sequence diagrams.
- Revise earlier diagrams based on new knowledge gained during this process.

# Copyright

