PassQueen
a.k.a dumpsqueen.com

CIW v5 Database Design Specialist

CIW 1D0-541

Version Demo

Total Demo Questions: 18

Total Premium Questions: 189

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Database Fundamentals	39
Topic 2, Database Design	63
Topic 3, Database Implementation	9
Topic 4, Database Programming	57
Topic 5, Database Administration	20
Topic 6, Database Connectivity	1
Total	189

QUESTION NO: 1

Which SQL statement is used to retrieve rows from one or more tables?

A. SELECT

B. CREATE

C. GRANT

D. COMMIT

ANSWER: A

Explanation:

`SELECT` is correct because it is the SQL statement used to query data stored in tables, views, or other queryable database objects. A SELECT statement can identify the columns to return, restrict rows with a WHERE clause, combine related tables with joins, group results, and order the output. Although SELECT is commonly grouped with Data Manipulation Language, it does not itself alter the stored data. Statements such as INSERT, UPDATE, and DELETE are used to add, change, or remove rows. See the PostgreSQL SELECT reference and the Microsoft SQL Server SELECT documentation.

QUESTION NO: 2

Consider the following relational algebraic expression: Which of the following SQL statements is equivalent to this relational algebraic expression?

A. SELECT*FROM Customers,Employees WHERE Sales_Rep_No = Cust_No;

B. SELECT Cust_No, Cust_Name, Emp_Name, Emp_Loc FROMCustomers, Employees WHERE Customers.Sales_Rep_No =Employees.Sales_Rep_No;

C. SELECT Cust_No, Cust_Name, Emp_Name, Emp_Loc FROMCustomers, Employees WHERE Employees.Sales_Rep_No =Customers.Sales_Rep_No;

D. SELECT * FROM Customers, Employees WHERECustomers.Sales_Rep_No = Employees.Sales_Rep_No;

ANSWER: B

Explanation:

`SELECT Cust_No, Cust_Name, Emp_Name, Emp_Loc FROM Customers, Employees WHERE Customers.Sales_Rep_No = Employees.Sales_Rep_No;` expresses a relational-algebra join between the `Customers` and `Employees` relations using the shared `Sales_Rep_No` value, followed by projection of precisely `Cust_No`, `Cust_Name`, `Emp_Name`, and `Emp_Loc`. The `FROM Customers, Employees` syntax forms the candidate row combinations, and the equality condition in `WHERE` retains the combinations in which a customer's sales-representative number matches an employee's sales-representative number. This is the conventional SQL representation of an equijoin when using comma join syntax. The select list is equally important: it implements projection by returning only the attributes named in the relational-algebra result, rather than every attribute from both source relations. Qualifying `Sales_Rep_No` with its relation name makes the comparison unambiguous and documents which instance of the identically named attribute is being compared. SQL join conditions and result-column selection are described in the PostgreSQL documentation on table expressions and joins; the role of the `SELECT` result list is described in the PostgreSQL SELECT reference.

QUESTION NO: 3

Which type of dependency exists when a non-key attribute depends on another non-key attribute?

A. Partial functional dependency

B. Transitive dependency

C. Referential dependency

D. Composite dependency

ANSWER: B

Explanation:

Transitive dependency is correct. A transitive dependency occurs when a key determines one non-key attribute, and that non-key attribute in turn determines another non-key attribute. For example, if *EmployeeID* determines *DepartmentID*, and *DepartmentID* determines *DepartmentName*, then *DepartmentName* is transitively dependent on *EmployeeID*. Such dependencies can create redundant data and update anomalies. Third normal form addresses this condition by separating the dependent facts into an appropriate related relation. See IBM documentation on normalization and Microsoft database normalization guidance.

QUESTION NO: 4

Consider the relation shown in the exhibit. Which of the following SQL statements would return a relation that excludes all customers with a Satisfaction_Rate of less than or equal to 80 unless the Sales_Office is located in Atlanta?

Customers Relation

Cust_No	Cust_Name	Satisfaction_Rate	Sales_Office	Sales_Rep_No
1011	MicroWidget	75	Atlanta	1350
1012	MacroWidget	90	New York	7403
1013	Xyz Corp	78	Los Angeles	2457
1014	DayCo	95	Atlanta	1350
1015	DigiTech	85	Chicago	3303
1016	DataTech	92	Los Angeles	2457
1017	UniSort	81	New York	7403

A. SELECT * FROM CustomersWHERE Satisfaction_Rate > 80OR Sales_Office = 'Atlanta';

B. SELECT * FROM CustomersWHERE Satisfaction_Rate <= 80AND Sales_Office = 'Atlanta';

C. SELECT * FROM CustomersWHERE Satisfaction_Rate >= 80;

D. SELECT * FROM CustomersWHERE Satisfaction_Rate >= 80AND NOT Sales_Office = 'Atlanta';

ANSWER: A

Explanation:

`SELECT * FROM Customers WHERE Satisfaction_Rate > 80 OR Sales_Office = 'Atlanta';` is correct because it directly expresses the required inclusion rule. A customer is returned when either condition is true: the customer has a Satisfaction_Rate greater than 80, or the customer belongs to the Atlanta sales office. Consequently, customers whose Satisfaction_Rate is 80 or lower are excluded unless their Sales_Office value is Atlanta. The `OR` operator is essential because Atlanta is an exception to the satisfaction-rate threshold, rather than an additional requirement. SQL evaluates a row against the complete Boolean predicate in the `WHERE` clause and returns the row when that predicate evaluates to true. The value `'Atlanta'` is written as a quoted character string literal, as required for textual comparison in standard SQL usage. This predicate also properly includes Atlanta customers regardless of whether their satisfaction rate is above, equal to, or below 80, which is exactly what “unless the Sales_Office is located in Atlanta” requires. See the PostgreSQL documentation on logical operators and the SQL Server documentation on SELECT and WHERE filtering.

QUESTION NO: 5

Consider the following relation definition:

STUDENT(

Student_Number: integer NOT NULL

Name: variable length character string length 20 NOT NULL)

Primary Key Student_Number

HOUSING(

Housing_ID: integer NOT NULL

Student_Number: integer NOT NULL

Building: variable length character string length 25 NOT NULL)

Primary Key Housing_ID

Foreign Key Student_Number References

STUDENT(Student_Number)

ON DELETE NO CHECK

ON UPDATE

Which integrity constraint is violated in this relation definition?

A. Entity integrity

B. Domain constraint

C. Referential integrity

D. Enterprise constraint

ANSWER: C

Explanation:

Referential integrity is violated because the HOUSING relation contains a foreign key, Student_Number, that links each housing record to a STUDENT record, but its delete/update referential action is not correctly defined. Referential integrity requires a foreign-key value to identify an existing candidate or primary-key value in the referenced relation. It also requires that changes to a referenced parent key be governed by a valid action so that the relationship remains consistent. A setting of *ON DELETE NO CHECK* would allow deletion without enforcing the relationship, potentially leaving HOUSING rows whose Student_Number values no longer identify a STUDENT row. In addition, the unfinished *ON UPDATE* clause does not provide a valid update action. Valid SQL referential actions normally include NO ACTION, RESTRICT, CASCADE, SET NULL, and SET DEFAULT, subject to database support. The relation should use an appropriate, complete action—such as ON DELETE CASCADE or ON DELETE NO ACTION and a defined ON UPDATE action—to preserve the required parent-child relationship. See the SQL Server documentation on primary and foreign key constraints and the PostgreSQL documentation on foreign-key constraints and referential actions.

QUESTION NO: 6

Consider the entity-relation (ER) diagram shown in the exhibit. When the logical database design phase is completed, which of the following is a valid DBDL description of the base relations for the ER diagram?

A. STUDENT(

B. STUDENT(

C. STUDENT(

D. STUDENT(

ANSWER: D

Explanation:

The intended correct answer is the fourth supplied `STUDENT(` choice, as indicated by the source question's existing answer designation. However, the JSON supplied for review is materially truncated: the exhibit is absent, and every answer choice ends immediately after `STUDENT(`. A valid Database Design Language description normally lists each base relation, its attributes, identifies primary-key attributes, and includes foreign keys created when relationships from the conceptual model are mapped to relations. The precise validity of a proposed relation set therefore depends on the missing entity attributes, identifiers, cardinalities, optionality, and any associative entities shown in the absent exhibit. In particular, relationship cardinality determines where foreign keys belong, while many-to-many relationships generally require their own relation. These are standard steps in converting an entity-relationship design into a logical relational schema. Because the complete text of the designated choice and the diagram were not included, its detailed DBDL syntax cannot be independently checked or reconstructed without inventing attributes and relationships. The answer designation is retained to preserve the available source answer while identifying this input limitation. See Oracle Database Concepts: Data Integrity and IBM documentation on relationship models.

QUESTION NO: 7

Which process is used to prevent the current database operation from reading or writing a data item while that data item is being accessed by another operation?

A. Lock

B. Deadlock

C. Time stamp

D. Transaction

ANSWER: A

Explanation:

Lock is correct because locking is a concurrency-control mechanism that a database management system uses to coordinate simultaneous operations on the same data item. When an operation acquires a lock on a row, table, page, or other database resource, the lock restricts conflicting access by other operations until the protected work is completed or the lock is released. This prevents interference such as one transaction modifying a value while another transaction attempts to read or modify that same value.

Locks can be applied in different modes according to the operation being performed. For example, an exclusive lock protects data being changed and prevents other operations from reading or writing it in ways that conflict with the update. A shared lock supports controlled reading while restricting incompatible modifications. By enforcing these rules, locking helps preserve transaction isolation and data consistency when many users or applications access the database concurrently. The database engine manages lock acquisition, compatibility, waiting, and release as part of transaction processing. Microsoft's overview of transaction locking and row versioning describes how locks protect resources and control concurrent access: SQL Server Transaction Locking and Row Versioning Guide. See also IBM's discussion of locks in transaction processing: Db2 locking.

QUESTION NO: 8

Which type of database anomaly can occur when a fact cannot be added to a relation until another unrelated fact is also known?

A. Deletion anomaly

B. Insertion anomaly

C. Concurrency anomaly

D. Referential anomaly

ANSWER: B

Explanation:

Insertion anomaly is correct. An insertion anomaly occurs when an improperly designed relation requires information about one subject to be supplied before information about another subject can be stored. For example, if employee and department data are combined in one relation, the organization may be unable to record a new department until at least one employee is assigned to it. Normalization separates facts about distinct entities into related relations, allowing each fact to be recorded independently when appropriate. This reduces unnecessary dependencies and improves the consistency of data maintenance. See Microsoft database normalization guidance and IBM documentation on normalization.

QUESTION NO: 9

The exhibit shows a relation for a company projects. Which candidate key(s) would best serve as the primary key for this relation?

Project Relation

Proj_ID	Item_Num	Item_Qty	Item_Price	S_Date	E_Date	Total_Cost
1001	3211	50	.70	2-2-99	2-2-00	3.50
1001	4311	100	.50	2-2-99	2-2-00	50.00
1002	3211	40	1.00	4-4-00	5-9-00	40.00
1003	5211	200	.50	5-5-00	7-8-00	100.00

A. S_Date and E_Date

B. ProjJD

C. Item_Num and E_Date

D. Proj_ID and Item_Num

ANSWER: D

Explanation:

Proj_ID and Item_Num is the appropriate candidate key because the two attributes together identify each row in the projects relation uniquely. A project identifier identifies the project, while an item number distinguishes the individual item, activity, or record associated with that project. The same item number can reasonably occur under different projects, and a project can contain multiple items; therefore, neither attribute alone supplies the row-level uniqueness required of a primary key. Used as a composite key, however, the pair expresses the relation's natural business identity: one specific item within one specific project.

A primary key must uniquely identify every tuple and must be minimal, meaning that no unnecessary attribute is included. The composite of **Proj_ID** and **Item_Num** satisfies both requirements when the table represents multiple items for each project. It also supports entity integrity, since primary-key values cannot be null, and provides a stable identifier that related tables can reference through foreign keys. This is consistent with relational database design guidance on primary keys and composite keys. See Oracle Database Concepts: Data Integrity and Microsoft: Description of database normalization basics.

QUESTION NO: 10

A large enterprise uses a two-tier database architecture and runs complex database applications.

Which term best describes the client in this system?

A. Fat client

B. Enterprise client

C. Thin client

D. Terminal client

ANSWER: A

Explanation:

Fat client is correct because, in a two-tier database architecture, the client commonly handles substantial application-processing responsibilities while communicating directly with the database server. A client in this arrangement typically contains the presentation interface and much or all of the business logic, submits SQL requests to the database server, and processes the returned results. When the enterprise runs complex database applications, the workstation needs enough local resources and installed software to execute that significant workload; this is the defining characteristic of a fat client. The database server remains responsible for centralized data storage, retrieval, integrity enforcement, transaction processing, and concurrency control, while the client performs the application-oriented processing associated with the user's work. This direct client-to-database-server model is the classic two-tier client/server design. Oracle's client/server documentation describes the client as the component that runs the application and requests database services, while the database server manages the database and fulfills those requests. See Oracle Database Client/Server Architecture. For a broader description of client/server application architecture and the division of client and server processing, see IBM client/server programming documentation.

QUESTION NO: 11

Which relational algebraic operation is used to select specific columns (attributes) from a relation?

A. Union

B. Difference

C. Projection

D. Intersection

ANSWER: C

Explanation:

Projection is the relational algebra operation used to choose specific attributes, or columns, from a relation. It produces a new relation containing only the named attributes, reducing the degree of the relation while retaining the relevant data values. Projection is conventionally represented by the Greek letter pi (π), with the desired attribute list written as a subscript; for example, $\pi_{EmployeeName,Department}$(Employee) returns only the *EmployeeName* and *Department* columns from the Employee relation. As a relational operation, projection also follows set semantics, so duplicate resulting tuples are eliminated. This is conceptually similar to specifying selected columns in the SELECT list of an SQL query, although SQL preserves duplicates unless DISTINCT is requested. Selecting columns is a fundamental database-design and query concept because it lets users retrieve only the attributes required for a report, view, or subsequent operation. IBM's overview of relational algebra identifies projection as the operation for selecting attributes from a relation, and the University of California, Berkeley's database materials describe projection as retaining specified columns: IBM Documentation: Relational algebra; UC Berkeley CS 186: Introduction to Database Systems.

QUESTION NO: 12

Which concurrency control method should be used only when conflicts between transactions rarely occur?

A. Locking

B. Time stamps

C. Optimistic

D. Serialization

ANSWER: C

Explanation:

Optimistic concurrency control is appropriate when transaction conflicts are expected to be rare. This approach lets transactions proceed without taking restrictive locks during their normal read and update work. Before a transaction commits, the database validates whether another transaction has changed data in a way that creates a conflict. If validation succeeds, the transaction is committed; if a conflict is detected, the transaction is rolled back or retried.

This method is efficient in environments with many reads, relatively independent updates, and low contention for the same rows or records. Avoiding routine locking reduces blocking and associated overhead, allowing multiple users to work concurrently with minimal interference. Its fundamental assumption is that most transactions will complete successfully on their first attempt. When collisions are uncommon, the occasional cost of retrying a failed transaction is lower than continuously coordinating access through locks. Microsoft describes optimistic concurrency as a method that checks whether data has changed since it was read before applying an update; see Entity Framework concurrency handling. The same validate-before-commit principle is also described in Microsoft SQL Server optimistic concurrency documentation.

QUESTION NO: 13

Which of the following best describes a composite key?

A. A composite key is a primary key that consists of the first two attributes of a relation.

B. A composite key is a primary or foreign key defined by its parent keys.

C. A composite key is a foreign key that consists of the same attributes as the primary key from a related table.

D. A composite key is a primary or foreign key that consists of two or more attributes of a relation.

ANSWER: D

Explanation:

A composite key is a key formed by combining two or more attributes (columns) of the same relation (table) so that they work together to identify a row or establish a relationship. The individual attributes in the combination do not necessarily identify a row uniquely on their own; it is their combined values that provide the required uniqueness or reference. Thus, “A composite key is a primary or foreign key that consists of two or more attributes of a relation.” is the best description. When used as a primary key, the combined columns uniquely identify each record in the table. When used as a foreign key, the combined columns reference the corresponding multi-column primary or unique key in a related table, preserving referential integrity. Composite keys are common in associative tables that implement many-to-many relationships, where the identifiers of the two related entities together identify each association. For example, an enrollment relation can use a student identifier plus a course identifier as its composite primary key, because the pair identifies one student’s enrollment in one course. Oracle’s SQL documentation describes a composite primary key as a key made from multiple columns, and PostgreSQL likewise documents that a primary key can consist of one or more columns. See Oracle Database SQL Language Reference: Constraints and PostgreSQL Documentation: Constraints.

QUESTION NO: 14

Which database security technique prevents invalid data from being entered into the database?

A. File locking

B. User authorization

C. Parity checks

D. Integrity controls

ANSWER: D

Explanation:

Integrity controls are the database security technique that prevents invalid data from being entered because they enforce rules governing what values and relationships the database may store. These controls preserve the accuracy, consistency, and reliability of data throughout inserts and updates. In a relational database, integrity controls commonly include data-type requirements, domain constraints, required-value rules, range or format checks, primary-key uniqueness, and foreign-key referential integrity. For example, a constraint can require an order quantity to be a positive number, prohibit duplicate customer identifiers, or require an order's customer identifier to match an existing customer record. If submitted data violates one of these rules, the database management system rejects the operation rather than storing invalid or inconsistent information. This is a core database-design and security principle: protecting data is not limited to restricting access; it also requires ensuring that accepted data conforms to defined business rules. Microsoft's documentation describes constraints as rules enforced on database columns and identifies primary-key, foreign-key, unique, check, and not-null constraints as mechanisms that maintain data integrity. See Microsoft Learn: Primary and foreign key constraints and Microsoft Learn: Unique and check constraints.

QUESTION NO: 15

Consider the following relational algebraic expression as well as the Employee and Department relations shown in the exhibit: Which of the following relations would result from the given relational algebraic expression?

A. OptionA

B. OptionB

C. OptionC

D. OptionD

ANSWER: D

Explanation:

OptionD is retained as the correct result because it is the answer identified as correct in the supplied question data. A relational-algebra result must be determined by applying the operator or operators in the displayed expression to the rows and attributes of the Employee and Department relations. For example, a selection retains qualifying tuples, a projection retains specified attributes, a join combines compatible tuples according to its predicate, and set operations require compatible relation schemas. The resulting relation's columns and rows therefore depend on the exact expression, predicates, attribute lists, and data values shown in the exhibit. In the JSON provided here, however, the relational-algebra expression and the Employee and Department relation contents are not included; the answer choices are also placeholder labels rather than displayed relations. Consequently, the supplied correct-answer designation is the only available basis for identifying the intended output. The underlying concepts are consistent with the relational model, in which a relation is a set of tuples described by attributes and queries derive result relations from source relations. See IBM's overview of the relational model and the relational operators.

QUESTION NO: 16

Consider the Dept1_Parts and Dept2_Parts relations shown in the exhibit. Which of the following SQL statements would create a set difference of the two relations with the widest variety of Structured Query Language dialects?

Part_ID	Part_Name	Description	Supp_ID
0312	bolt	hexagon bolt	221
0322	screw	capscrew	441
0332	socket screw	button head	551
0342	flange	blind flange	331
0352	socket screw	countersunk	441

Dept1_Parts Relation

Part_ID	Part_Name	Description	Supp_ID
0302	flange	slip-on flange	331
0322	screw	capscrew	441
0332	socket screw	button head	551
0362	bolt	studbolt	441

Dept2_Parts Relation

A. SELECT *
FROM Dept1_Parts
EXCEPT
(SELECT Part_ID
FROM Dept2_Parts);

B. SELECT *
FROM Dept1_Parts
MINUS
(SELECT Part_ID
FROM Dept2_Parts);

C. SELECT *
FROM Dept1_Parts
DIFFERENCE
(SELECT Part_ID
FROM Dept2_Parts);

D. SELECT *
FROM Dept1_Parts
DIFFERENCE
(SELECT Part_ID
FROM Dept2_Parts);

ANSWER: A

Explanation:

`SELECT * FROM Dept1_Parts EXCEPT (SELECT Part_ID FROM Dept2_Parts);` expresses a set difference: it returns the rows produced by the first query that are not present in the result of the second query. In this case, it identifies part identifiers listed for Dept1_Parts but absent from Dept2_Parts. `EXCEPT` is the SQL-standard set-difference operator, making it the appropriate choice when the goal is broad SQL dialect compatibility among set operators. The two component queries must return the same number of columns in corresponding, compatible data types; as shown, this assumes Dept1_Parts contains the displayed Part_ID column as its selected value. Set operators also normally apply distinct-set semantics, so duplicate matching values are not returned repeatedly. PostgreSQL documents `EXCEPT` as returning rows in the first query result but not the second, and Microsoft SQL Server likewise documents it as the operator for distinct rows returned by the left query that are not output by the right query. See the PostgreSQL documentation for UNION, INTERSECT, and EXCEPT and Microsoft documentation for EXCEPT.

QUESTION NO: 17

A foreign key maps to a:

A. Prime key

B. Indirect key

C. Parent key

D. Composite key

ANSWER: C

Explanation:

Parent key is correct. A foreign key is a column, or a set of columns, in one table whose values reference a key in a related parent table. This relationship establishes a link between the child table and the parent table, allowing related records to be joined reliably. In most database designs, the referenced parent key is the table's primary key; in systems that support it, it can also be another key constrained to contain unique values. The foreign-key values in the child table must correspond to an existing value in that referenced parent key, unless the foreign-key column is permitted to contain NULL. This rule is called referential integrity and prevents child records from pointing to nonexistent parent records. For example, an Orders table can contain a CustomerID foreign key that maps to the CustomerID key in a Customers table. Database platforms document this relationship explicitly as a foreign key referencing a primary or unique key in the parent table. See Microsoft Learn: Create foreign key relationships and Oracle Database Concepts: Integrity Constraints.

QUESTION NO: 18

Consider the Registration relation shown in the exhibit. Which of the following SQL statements would return the Registration2 relation from the Registration relation?

Registration_ID	Student_ID	Course_Code	First_Name	Last_Name
1001	S320	M3455	Teri	Chan
1002	S255	M3455	Carlos	Trujillo
1003	S511	A4343	Helen	Yang
1004	S812	S4511	Robert	Cray
1005	S320	A4343	Teri	Chan
1006	S255	M4422	Carlos	Trujillo
1007	S511	M4433	Helen	Yang
1008	S812	S2212	Robert	Cray

Registration Relation

1003	S511	A4343	Helen	Yang
1005	S320	A4343	Teri	Chan

Registration2 Relation

A. FROM Registration;

B. Registration WHERE

C. WHERE Course_Code ='A4343';

D. SELECT Registration_ID, Student_ID, First_Name, Last_Name

ANSWER: C

Explanation:

`WHERE Course_Code ='A4343';` is the clause that produces the Registration2 relation by restricting the rows from Registration to those whose `Course_Code` value is A4343. In relational terms, this is a selection operation: the resulting relation retains the attributes supplied by the surrounding `SELECT` statement but includes only tuples satisfying the stated predicate. Thus, when used as part of a complete query such as `SELECT Registration_ID, Student_ID, First_Name, Last_Name FROM Registration WHERE Course_Code = 'A4343';`, it retrieves the registrations associated with that course and yields the filtered relation shown in the exhibit. SQL evaluates a `WHERE` search condition for each candidate row and returns rows for which that condition is true. The equality predicate is appropriate because the target course is identified by one specific course-code value. The SQL standard convention is to delimit character literals with single quotation marks; the quotation mark characters in the displayed text should therefore be interpreted as standard single quotes in an executable SQL statement. See the PostgreSQL SELECT documentation and the Microsoft SELECT documentation for the role of the `WHERE` clause in filtering result rows.