

1NF – A relvar is in 1NF if and only if, in every legal value of that relvar, every tuple contains exactly one value of each attribute.

2NF – A relvar is in 2NF if and only if it is in 1NF and every nonkey attribute is irreducibly dependent on the primary key.

3NF – A relvar is in 3NF if and only if it is in 2NF and every nonkey attribute is nontransitively dependent on the primary key.

4NF – Relvar R is in 4NF if and only if, whenever there exist subsets A and B of the attributes of R such that the nontrivial MVD R →> B is satisfied, then all attributes of R are functionally dependent on A. Note: The MVD A →> B is trivial if either A is a superset of B or the union AB of A and B is the entire heading.

5NF – A relvar R is in 5NF – also called **projection-join normal form (PJ/NF)** – if and only if every nontrivial join dependency that is satisfied by R is implied by the candidate key(s) of R where the join dependency * {A, B, ..., Z} on R is trivial if and only if at least one of A, B, ..., Z is the set of all attributes of R and the join dependency * {A, B, ..., Z} on R is implied by the candidate keys of R if and only if each of A, B, ..., Z, is a superkey of R.

B/CNF – A relvar is in BCNF if and only if every nontrivial, leftirreducible FD has a candidate key as its determinant.

Benefits of DB approach – the data can be shared, redundancy can be reduced, inconsistency can be avoided, transaction support can be provided, integrity can be maintained, security can be enforced, conflicting requirements can be balanced, standards can be enforced.

bill-of-material – the rel. that certain parts include other parts as immediate components

BUILD-IN TYPES: BOOLEAN, BIT [VARYING] (n), BINARY LARGE OBJECT (n), CHARACTER [VARYING] (n), CHARACTER LARGE OBJECT (n), NUMERIC (p,q), DEIMAL (p,q), INTEGER, SMALLINT, FLOAT (p), DATE, TIME, TIMESTAMP, INTERVAL

CELLAR – SQL files called **tables (relations)**. **Rows (tuples)** → records, **Columns (attributes)** → fields

Conceptual view – representation of the entire information content of the db. Consists of many types of **conceptual records** and is defined by means of the **conceptual schema**

Create – create a new table (CREATE TABLE PART_STRUCTURE (MAJOR_P# P#, MINOR_P# P#, QTY QTY, PRIMARY KEY (MAJOR_P#, MINOR_P#), FOREIGN KEY (MAJOR_P#) REFERENCES PART, FOREIGN KEY (MINOR_P#) REFERENCES PART);

DA → makes strategic and policy decisions; **DBA** → provides technical support for implementing them.

Data communication manager – software component that controls message transmissions(not part of the DBMS)

Data dictionary – data about the data

Data independ. → immunity of application programs to changes in the way the data is physically stored and accessed.

Data model – an abstract, self-contained logical definition of the objects, operators, and so forth, that together constitute the abstract machine with which users interact.

Data sublanguage(DSL) – subset of the total language that is concerned specifically with db obj. and operations. It is a combination of DDL(definitions) and DMLprocessing

Databas – collection of persistent data that is used by the application systems of some given enterprise.

Database system – Comput. System whose overall purpose is to store information and to allow users to retrieve and update that information on demand.(data, hardware, software -dbms)

DBMS: the soft that handles all access to the DB.

Delete – Delete data from a table (DELETE FROM SP WHERE P# = P# ('P2')); DELETE S WHERE CITY = 'Paris' ;)

Denormalization – replacing a set of R1, R2, ..., Rn relvars by their join R, such that for all i(i=1,2,3...n) projecting R over the attributes of Ri is guaranteed to yield Ri again.

Difference – given two relations a and b of the same type, the difference between those two relations, a MINUS b, is a relation of the same type, with body consisting of all tuples t such that t appears in a and not b

Distinct type – CREATE TYPE WEIGHT AS DECIMAL (5,1) FINAL;

Division – division of a by b per c (where a is the dividend, b is the divisor, and c is the mediator is a relation with heading X and body consisting of all tuples {X x} appearing in a such that a tuple { Xx, Y y} appears in c for all tuples { Y y } appearing in b. The result consists of those X values from a whose corresponding Y values in c include all Y values from b, loosely speaking.

Entitle – any distinguishable object that is to be represented in the DB.

Exclusive lock: (X) write lock

External view – individual user's view. Consists of many types of **external record**. It is defined by means of

external schema – consists of definitions of each of the various ext. record types in that ext. view.

Foreign key – a set of attributes of some relvar R2 whose values are required to match values of some candidate key of some relvar R1

Functional Dependency - Let r be a relation, and let X and Y be arbitrary subsets of the set of attributes of r. Then we say that Y is functionally dependent on X if and only if each X value in r has associated with it precisely one Y value in r.

Implementation – a physical realization on a real machine of the components of the abstract machine that together constitute that model.

Insert - Insert new data into a table (INSERT INTO TEMP (P#, WEIGHT) SELECT P#, WEIGHT FROM P WHERE COLOR = COLOR ('Red');

Integrated data – DB can be thought of as a unification of several otherwise distinct files with any redundancy among those files partly or wholly eliminated.

Internal view – low-level representation of the entire db. Consists of many occurrences of many types of **internal record** and is described by means of **internal schema** which defines the various stored record types and specifies what indexes exist, how stored fields are represented, what physical sequence the stored records are in and so on.

Intersect – given two relations a and b of the same type, the intersection of those two relations, a INTERSECT b, is a relation of the same type, with body consisting of all tuples t such that t appears in both a and b

Join - combines two tables into one on the basis of common values in a common column. SELECT S.S# , SNAME, STATUS, City, P#, QTY FROM S, SP WHERE S.S# = SP.S#;

Mappings: conceptual/internal → how conceptual records and fields are represented at the int. level;

external/conceptual → logical data independence; external/external → external views defined by other external views.

Nonscalar type – type whose values are explicitly defined to have a set of user-visible, directly accessible components.

Normalization – normalizing a relvar R means replacing R by a set of projections R1, R2, ..., Rn, such that R is equal to the join of R1, R2, ..., Rn; the objective is to reduce redundancy, by making sure that each of the projections R1, R2, ..., Rn is at the highest possible level of normalization.

normalization procedure: Take projections of the original 1NF relvar to eliminate Fds that are not irreducible. Take projections of the produced 2NF relvars to eliminate transitive Fds that are not irreducible, take projections of the produced 3NF relvars to eliminate remaining Fds in which the determinant is not a candidate key. Take projections of

the produced BCNF relvars to eliminate MVDs that are not also Fds. Take projections of the produced 4NF relvars to eliminate Jds that are not implied by the candidate key if you can find them.

Operations – adding, inserting, retrieving, deleting, changing, removing

Operators – select, insert, delete and update (change).

Optimizer – determine an efficient way of implementing the request.

Outer join: an extended form of the regular or **inner join** operation. It differs from the inner join in that tuples in one relation having no counterpart in the other join in the result with nulls in the other attribute position, instead of simply being ignored as they normally are.

Planned request – one for which the need was redoesen

Primary key → no two tuples ever contain the same value

Product – a set of ordered tuples

Project – extracts specified columns from a table SELECT S#, CITY FROM S;

Project – the projection of relation a on X, Y, ..., Z – is a relation with a heading derived from the heading of a by removing all attributes not mentioned in the set {X, Y, ..., Z}, and a body consisting of all tuples { X x, Y y, ..., Z z } such that a tuple appears in a with X value x, Y value y, ..., and Z value z

Query language processor – the build-in app by which user can issue DB requests

Relation – mathematical term for table; A relation value r consists of heading and body where the heading of r is a tuple heading and the body r is a set of tuples all having the same heading.

Relational algebra – a collection of operations on relations (union, intersect, difference, product, restrict, project, join and devide)

Relational model of data – data is represented by means of rows in tables, and such rows can be directly interpreted as true propositions; Operators are provided for operating on rows in tables and those operators directly support the process of inferring additional true propositions from the given ones;

Restrict – extracts specified rows from a table (sometimes called **select**) (SELECT S#,P#,QTY FROM SP WHERE QTY < QTY (150) ;

Restrict – restriction of relation a on attributes X and Y – a WHERE X =/= <, < etc Y – is a relation with the same heading as a and with body consisting of all tuples of a such that the above expression evaluates to TRUE for the tuples in question.

Row – sql does not support tuples, as such, at all; instead it supports rows, which have left-to-right ordering to their components. Within a given row, the component values – which are called column values if the row is immediately contained in a table, or field values otherwise – are thus identified primarily by their ordinal position.

Rules for MVDs : Complementation: if A, B, and C together include all attributes of the relvar and A is a superset of B ∩ C, then A →> B if and only if A →> C; **Reflexivity:** if B is a subset of A, then A →> B;

Augmentation: If A →> B and C is a subset of D, then AD →> BC; **Transitivity:** If A →> B and B →> C, and A →> C-B; **Pseudotransitivity:** If A →> B and BC →> D, then AC →> D – BC; **Union:** if A →> B, and A →> C, then A →> BC; **Decomposition:** If A →> BC, then A →> B ∩ C, A →> B-C, and A →> C-B;

Replication: if A →> B, then A →> B; **Coalescence:** If A →> B and C →> D and D is a subset of B and B ∩ C is empty, then A →> D.

Scalar type – one that is not scalar.

Serializability: A given execution of a given set of transactions is serializable if and only if it is equivalent to some serial execution of the same transactions, where a serial execution on e in which the transactions are run one at a time in some sequence.

Shared data – DB can be shared among different users, in the sense that different users can have access to the same data, possibly even at the same time(“concurrent access”)

Shared lock:(S) read lock

Stored field – the smallest unit of stored data.

Stored file – collection of all currently existing occurrences of one type of stored record.

Stored record - collection of related stored fields.

Table – sql does not support relations, as such, at all; instead it supports tables. The body of a table in SQL is not a set of tuples but a bag(multiset) of rows instead; thus, the columns of such a table have a left-to-right ordering, and there can be duplicate rows.

Table heading – a set of column-name-type-name pairs

The left and the right sides of an FD are called the **determinant** and the **dependent**.

Three levels of architecture: **internal** (storage) level – physical storage; **external** (user logical) level – closest to the users; **conceptual** (community logical) – indirection between first two

Transaction – a logical unit of work, typically involving several db operations.

Transaction – logical unit of work typically involving several database operations.It is atomic – they are either executed or not executed at all. Also they are durable – once a transaction successfully executes COMMIT, its updates are guaranteed to appear in the DB. They are isolated – the DB updates made by T1 are not mae visible to any distinct transaction T2 until and unless T1 successfully executes COMMIT. They are serializable – produce the same result as executing those same transactions one at a time in some unspecified serial order.

Transaction manager – enforces certain revocery and concurrency controls

Trigger – triggers consist of three parts, an event (operation on the database), condition (a boolean expression that has to evaluate to TRUE in order for the action to be executed and action is the triggered procedure proper

Tuple – a tuple value of a given collection types Ti (i = 1,2,3 ...) is a set of ordered triples of the form <Ai, Ti vi>, where Ai is an attribute name, Ti is a type name and vi is a value of type Ti and the value ni is the degree or arity of ti, the ordered triple <Ai, Ti, vi> is a component of t, the ordered pair <Ai, Ti> is an attribute of t, and it is uniquely identified by the attribute name Ai, the complete set of attributes is the heading of t.

Union – the set of all elements belonging to either or both given sets

Unplanned request - an ad hoc query or update that is a request for which the need was not seen in advance

Update – Modify data in a table (UPDATE S SET STATUS= 2 * STATUS , CITY = ' Rome' WHERE CITY = 'Paris' ;

Utilities – load routines(create initial versions of db from regular files); unload/reload – dump/restore – routines (backup); reorganization routines (rearrange data for performance);statistical routines(performance stats); analysis routines (analyze stats)

Value – an individual constant

Variable – holder for an appearance of a value

View – a named expression of the relational algebra VAR GOD_SUPPLIER VIEW (S WHERE STATUS > 15) { S#, STATUS, CITY }

1NF	first normal form	ANSI	American National Standards Institute	CLOB	character large object	DDB	distributed DBMS
2NF	second normal form			CNF	conjunctive normal form	MS	
2PC	two-phase commit	ANSI/SPARC	literally, ANSI/Systems Planning and Requirements Committee;	COD	literally, Conference on Data Systems Languages; used to refer to certain prerelational (actually network) systems such as IDMS	DDL	data definition language
2PL	two-phase locking			ASYL	used to refer to the three-level database system architecture described in Chapter 2	DES	Data Encryption Standard
2VL	two-valued logic					DK/N	domain-key normal form
20C	same as 2PC					F	
20L	same as 2PL			CPU	central processing unit	DML	data manipulation language
3GL	third-generation language	API	application programming interface	CS	cursor stability (DB2)	DNF	disjunctive normal form
3NF	third normal form			CWA	Closed World Assumption	DOM	Document Object Model (XML)
3VL	three-valued logic			DA	data administrator	DRDA	Distributed Relational Database Architecture (IBM)
4GL	fourth-generation language	ARIE	Algorithms for Recovery and Isolation	DB/D	database/data communications	DSL	data sublanguage
4NF	fourth normal form	S	Exploiting Semantics	C		DSS	decision support system
4VL	four-valued logic	AST	automatic summary table	DBA	database administrator	DTD	Document Type Definition (XML)
5NF	fifth normal form (same as PJ/NF)	BB	same as GB	DBM	database management system	DUW	distributed unit of work
6NF	sixth normal form	BCNF	Boyce/Codd normal form	S		E/R	entity/relationship
A	ALGEBRA	BLOB	binary large object	&D	<i>Database Programming & Design</i> (originally a hardcopy magazine; later online at http://www.dbpd.com ; superseded by <i>Intelligent Enterprise</i>)	EB	same as XB
ACID	atomicity-consistency-isolation-durability	BNF	Backus-Naur form; Backus normal form			ECA	event-condition-action
ACM	Association for Computing Machinery	CAC	<i>Communications of the ACM</i> (ACM publication)	DBT	literally, Data Base Task Group; used interchangeably in database contexts with CODASYL	EDB	extensional database
ADT	abstract data type	CAD/	computer-aided design/computer-aided manufacturing	G		EDI	Electronic Data Interchange
AES	Advanced Encryption System	CAM	computer-aided software engineering			EKNF	elementary key normal form
ALGEBRA	A Logical Genesis Explains Basic Relational Algebra	CASE	computer-aided software engineering			EMV	embedded MVD
		CDO	class-defining object	DC	data communications	D	
		CIM	computer-integrated manufacturing	DCO	"domain check override"	EOT	end of transaction
		CLI	Call-Level Interface	DDB	distributed database	FD	functional dependence

FLOW R	<i>for-let-where-order-by-return</i> (XML)	JDBC	"Java Database Connectivity" (officially just a name, not an abbreviation for anything at all)	PRTV PSM	Peterlee Relational Test Vehicle Persistent Stored Modules (part of the SQL standard)	TODS	<i>Transactions on Database Systems</i> (ACM publication)
FTP	File Transfer Protocol (usually "ftp," all lowercase)	K	1024 (sometimes 1000)	PSVI	Post Schema Validation Infoset (XML)	TP	transaction processing
GB	gigabyte (1024MB)	KB	kilobyte (1024 bytes)	QBE	Query-By-Example	TR	Transaction Processing Council
GIS	geographic information system	LAN	local area network	QUEL	Query Language (Ingres)	U	update (lock)
GML	Generalized Markup Language	LOB	large object	RAID	redundant array of inexpensive disks	UDF	user-defined function
HOL	hybrid OLAP	LSP	Liskov Substitution Principle	RDA	Remote Data Access	UDO	user-defined operator
AP		MB	megabyte (1024KB)	RDB	relational database	UDT	user-defined type
HTM	HyperText Markup Language	MLS	multi-level secure	RDBMS	relational DBMS	UML	Unified Modeling Language
L		MOLAP	multi-dimensional OLAP	RID	record ID; row ID	<i>unk</i>	<i>unknown</i> (truth value)
HTTP	Hypertext Transfer Protocol (usually "http," all lowercase)	MQT	materialized query table	RR	read-read; repeatable read (DB2)	UNK	unknown (null)
I/O	input/output	MVD	multi-valued dependence	RSA	Rivest-Shamir-Adelman (encryption method)	UOW	unit of work
IDB	intensional database	NCITS	<i>see</i> INCITS	RUW	remote unit of work	URI	Uniform Resource Identifier
IDMS	Integrated Database Management System	NCITS/H2	<i>see</i> INCITS/H2	RVA	relation-valued attribute	URL	Uniform Resource Locator
IFIP	International Federation for Information Processing	NF ²	"NF squared" = NFNF = non first normal form (?)	S	shared (lock)	VLDB	very large database; Very Large Data Bases (annual conference)
IEEE	Institute for Electrical and Electronics Engineers	ODBC	Open Database Connectivity	SGML	Standard GML	VSAM	Virtual Storage Access Method
IMS	Information Management System	ODMG	Object Data Management Group	SIGMOD	Special Interest Group on Management of Data (ACM special interest group)	W3C	World Wide Web Consortium
INCITS	ANSI International Committee on Information Technology Standards (formerly called NCITS, and before that X3)	ODS	operational data store	SOAP	Simple Object Access Protocol	WAL	write-ahead log
S		OIDS	object ID	SPARC	<i>see</i> ANSI/SPARC	WAN	wide area network
INCITS	INCITS database committee	OLAP	online analytic processing	SQL	(originally) Structured Query Language; sometimes Standard Query Language; officially just a name, not an abbreviation for anything at all	WFF	well-formed formula
S/H2		OLCP	online complex processing	SQL/MM	SQL/Multimedia	WORM	write once/read many times
IND	inclusion dependence	OLDM	online decision management	SVG	Scalable Vector Graphics	WWW	World Wide Web (usually "www," all lowercase)
IS	intent shared (lock); information systems	OLTP	online transaction processing	TB	terabyte (1024GB)	WYSIWYG	what you see is what you get
ISBL	Information System Base Language (PRTV)	OMG	Object Management Group	TCB	Trusted Computing Base	X	exclusive (lock)
ISO	International Organization for Standardization	OO	object-oriented; object orientation	TCP/IP	Transmission Control Protocol/Internet Protocol	X3	<i>see</i> INCITS
IT	information technology	OODB	object-oriented database (= object database)	TID	tuple ID	X3H2	<i>see</i> INCITS/H2
IX	intent exclusive (lock)	OODBMS	object-oriented DBMS (= object DBMS)			XB	exabyte (1024PB)
JACM	<i>Journal of the ACM</i> (ACM publication)	OOP	object-oriented programming language (= object programming language)			XML	Extensible Markup Language
JD	join dependence	OQL	Object Query Language (part of ODMG proposal)			XQuery	XML Query
		OSI	Open Systems Interconnection			XQL	XML Query Language (not the same as XQuery)
		OSQL	"Object SQL"			XSL	XML Stylesheet Language
		PB	petabyte (1024TB)			XSLT	XML Stylesheet and Transformation Language
		PC	personal computer			YB	yottabyte(1024ZB)
		PJ/NF	projection-join normal form			ZB	Zettabyte(1024XB)
		PODS	Principles of Database Systems (ACM conference)				

store_name	Sales	Date
Los Angeles	\$1500	Jan-05-1999
San Diego	\$250	Jan-07-1999
Los Angeles	\$300	Jan-08-1999
Boston	\$700	Jan-08-1999

```

SELECT "column_name" FROM "table_name" (SELECT store_name FROM Store_Information)
SELECT DISTINCT "column_name" FROM "table_name" (SELECT DISTINCT store_name FROM Store_Information)
SELECT "column_name" FROM "table_name" WHERE "condition" (SELECT store_name FROM Store_Information WHERE Sales > 1000)
SELECT "column_name" FROM "table_name" WHERE "simple condition" [(AND/OR) "simple condition"]+
(SELECT store_name FROM Store_Information WHERE Sales > 1000 OR (Sales < 500 AND Sales > 275)
SELECT "column_name" FROM "table_name" WHERE "column_name" IN ('value1', 'value2', ...)
(SELECT * FROM Store_Information WHERE store_name IN ('Los Angeles', 'San Diego'))
SELECT "column_name" FROM "table_name" WHERE "column_name" BETWEEN 'value1' AND 'value2' (SELECT * FROM Store_Information WHERE Date BETWEEN 'Jan-06-1999' AND 'Jan-10-1999')
SELECT "column_name" FROM "table_name" WHERE "column_name" LIKE ('PATTERN') (SELECT * FROM Store_Information WHERE store_name LIKE '%AN%')
SELECT "column_name" FROM "table_name" [WHERE "condition"] ORDER BY "column_name" [ASC, DESC]
(SELECT store_name, Sales, Date FROM Store_Information ORDER BY Sales DESC)
SELECT SUM(Sales) FROM Store_Information
SELECT COUNT(DISTINCT store_name) FROM Store_Information
SELECT "column_name1", SUM("column_name2") FROM "table_name" GROUP BY "column_name1" (SELECT store_name, SUM(Sales) FROM Store_Information GROUP BY store_name)
SELECT A1.region_name REGION, SUM(A2.Sales) SALES FROM Geography A1, Store_Information A2 WHERE A1.store_name = A2.store_name GROUP BY A1.region_name
[SQL Statement 1] UNION [SQL Statement 2] (SELECT Date FROM Store_Information UNION SELECT Date FROM Internet_Sales)
CREATE TABLE "table_name" ("column 1" "data_type_for_column_1", "column 2" "data_type_for_column_2", ...)
(CREATE TABLE customer (First_Name char(50), Last_Name char(50), Address char(50), City char(50), Country char(25), Birth_Date date)
TABLE Customer (First_Name char(50), Last_Name char(50), Address char(50), City char(50), Country char(25), Birth_Date date)
CREATE VIEW V_Customer AS SELECT First_Name, Last_Name, Country FROM Customer
CREATE VIEW V_REGION_SALES AS SELECT A1.region_name REGION, SUM(A2.Sales) SALES FROM Geography A1, Store_Information A2 WHERE A1.store_name = A2.store_name GROUP BY A1.region_name)
CREATE TABLE Customer (SID integer, Last_Name varchar(30), First_Name varchar(30), PRIMARY KEY (SID));
CREATE TABLE Order (Order_ID integer, Order_Date date, Customer_SID integer, Amount double, Primary Key (Order_ID), Foreign Key (Customer_SID) references Customer(SID));
ALTER TABLE orders ADD FOREIGN KEY (customer_sid) REFERENCES customer(sid);
DROP TABLE customer, TRUNCATE TABLE customer;
INSERT INTO "table_name" ("column1", "column2", ...) VALUES ("value1", "value2", ...) (INSERT INTO Store_Information (store_name, Sales, Date) VALUES ('Los Angeles', 900, 'Jan-10-1999'))
INSERT INTO Store_Information (store_name, Sales, Date) SELECT store_name, Sales, Date FROM Sales_Information WHERE Year(Date) = 1998
ALTER TABLE table_name ADD [COLUMN] column_declare
ALTER TABLE table_name ADD constraint_declare
ALTER TABLE table_name DROP [COLUMN] column_name
ALTER TABLE table_name DROP CONSTRAINT constraint_name
ALTER TABLE table_name DROP PRIMARY KEY
ALTER TABLE table_name ALTER [COLUMN] column_name SET default_expr
ALTER TABLE table_name ALTER [COLUMN] column_name DROP DEFAULT
DROP TABLE [ IF EXISTS ] table_name1, table_name2, ...
CREATE VIEW table_name [ ( column_name1, column_name2, ... ) ] AS SELECT
DROP VIEW table_name
CREATE SEQUENCE name [ INCREMENT increment_value ] [ MINVALUE minimum_value ] [ MAXVALUE maximum_value ] [ START start_value ] [ CACHE cache_value ] [ CYCLE ]
DROP SEQUENCE name

```

```

COMPACT TABLE table_name
INSERT INTO table_name [ ( col_name1, col_name2, ... ) ] VALUES ( expression1_1, expression1_2, ... ), ( expression2_1, expression2_2, ... ), ...
INSERT INTO table_name [ ( col_name1, col_name2, ... ) ] SELECT ...
INSERT INTO table_name SET col_name1 = expression1, col_name2 = expression2, ...
DELETE FROM table_name [ WHERE expression ] [ LIMIT limit_amount ]
UPDATE table_name SET col_name1 = expression1, col_name2 = expression2, ... [ WHERE expression ] [ LIMIT limit_amount ]
SELECT [ DISTINCT ] ALL ] column_expression1, column_expression2, ... [ FROM from_clause ] [ WHERE where_expression ] [ GROUP BY expression1, expression2, ... ] [ HAVING having_expression ] [ ORDER BY order_column_expr1, order_column_expr2, ... ]
CREATE USER username SET PASSWORD 'password' [ SET GROUPS groups_list ] [ SET ACCOUNT ( LOCK | UNLOCK ) ]
ALTER USER username SET PASSWORD 'password' [ SET GROUPS groups_list ] [ SET ACCOUNT ( LOCK | UNLOCK ) ]
DROP USER username
GRANT privileges ON database_object TO ( PUBLIC | user_list ) [ WITH GRANT OPTION ]
REVOKE [ GRANT OPTION FOR ] privileges ON database_object FROM ( PUBLIC | user_list )
privileges ::= priv_item1, priv_item2, ...
priv_item ::= ALL | PRIVILEGES ] | SELECT | INSERT | UPDATE | DELETE | REFERENCES | USAGE
database_object ::= [ TABLE ] table_name | SCHEMA schema_name
user_list ::= PUBLIC | username1, username2, ...
SET variable = expression
SET AUTO COMMIT ( ON | OFF )
SET TRANSACTION ISOLATION LEVEL ( SERIALIZABLE )
SET SCHEMA schema_name
DESCRIBE table_name
SHOW engine_variable (engine_variable ::= TABLES | SCHEMA | STATUS | CONNECTIONS)
SHUTDOWN
JOIN: SELECT First_Name, Last_Name, ISBN FROM Author INNER JOIN Book_Author ON Author.Author_ID = Book_Author.Author_ID;

```