



Most permission statements have the format:

AUTHORIZATION PERMISSION ON SECURABLE:NAME TO PRINCIPAL

AUTHORIZATION must be GRANT, REVOKE or DENY.

- **PERMISSION** is listed in the charts below.
- **ON SECURABLE:NAME** is the server, server object, database, or database object and its name. Some permissions do not require **ON SECURABLE:NAME**.
- **PRINCIPAL** is the login user, or role which receives or loses the permission. Grant permissions to roles whenever possible.

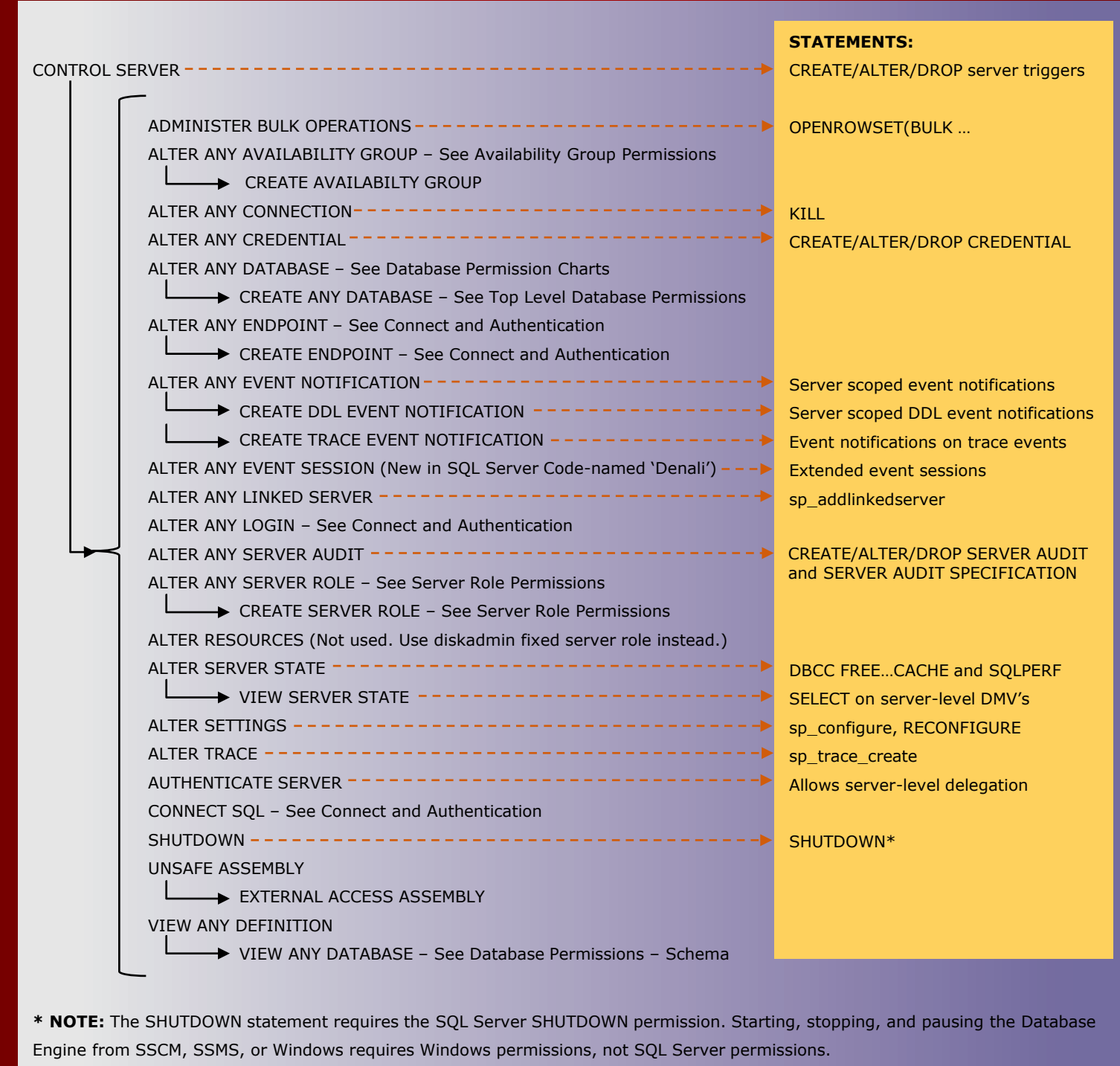
Some grant statements: GRANT UPDATE ON OBJECT:production.Parts TO PartsTeam

Denying a permission at any level, overrides a related grant.

To remove a previously granted permission, use REVOKE, not DENY.

- Most of the more granular permissions are included in more than one higher level scope permission. So permissions can be inherited from more than one type of higher scope.
- Black, green, and blue arrows and boxes point to subordinate permissions that are included in the scope of higher a level permission.
- Brown arrows and boxes indicate some of the statements that can use the permission.

Top Level Server Permissions



CONTROL SERVER → **CONTROL ON LOGIN::<name>**

VIEW ANY DEFINITION → **VIEW DEFINITION ON LOGIN::<name>**

ALTER ANY LOGIN → **ALTER ON LOGIN::<name>**

CONNECT SQL → **CONNECT SQL**

STATEMENTS: EXECUTE AS

- ALTER LOGIN, sp_addlinkedsevrlogin
- DROP LOGIN
- CREATE LOGIN

Notes:

- The CREATE LOGIN statement creates a login and grants CONNECT SQL to that login;
- Enabling a login (ALTER LOGIN <name> ENABLE) is not the same as granting CONNECT SQL permission
- To map a login to a credential, see ALTER ANY CREDENTIAL.
- When contained databases are enabled, users can access SQL Server without a login. See database user permissions.
- To connect using a login you must have :
 - o An enabled login
 - o CONNECT SQL
 - o CONNECT for the database (if specified)

CONTROL ON ENDPOINT::<name>

VIEW ANY DEFINITION → **VIEW DEFINITION ON ENDPOINT::<name>**

ALTER ANY ENDPOINT → **ALTER ON ENDPOINT::<name>**

CREATE ENDPOINT → **CREATE ENDPOINT**

STATEMENTS:

- ALTER ENDPOINT
- DROP ENDPOINT
- CREATE ENDPOINT

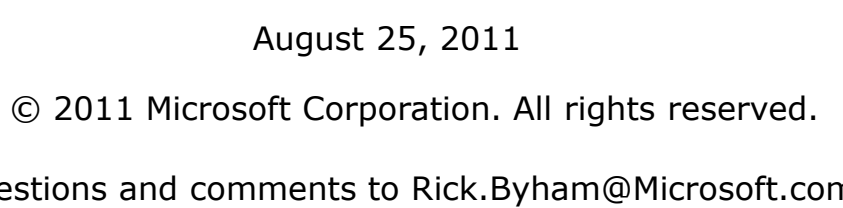
[illegible]

The diagram illustrates the sequence of operations for creating an availability group on a secondary server. It shows the following steps:

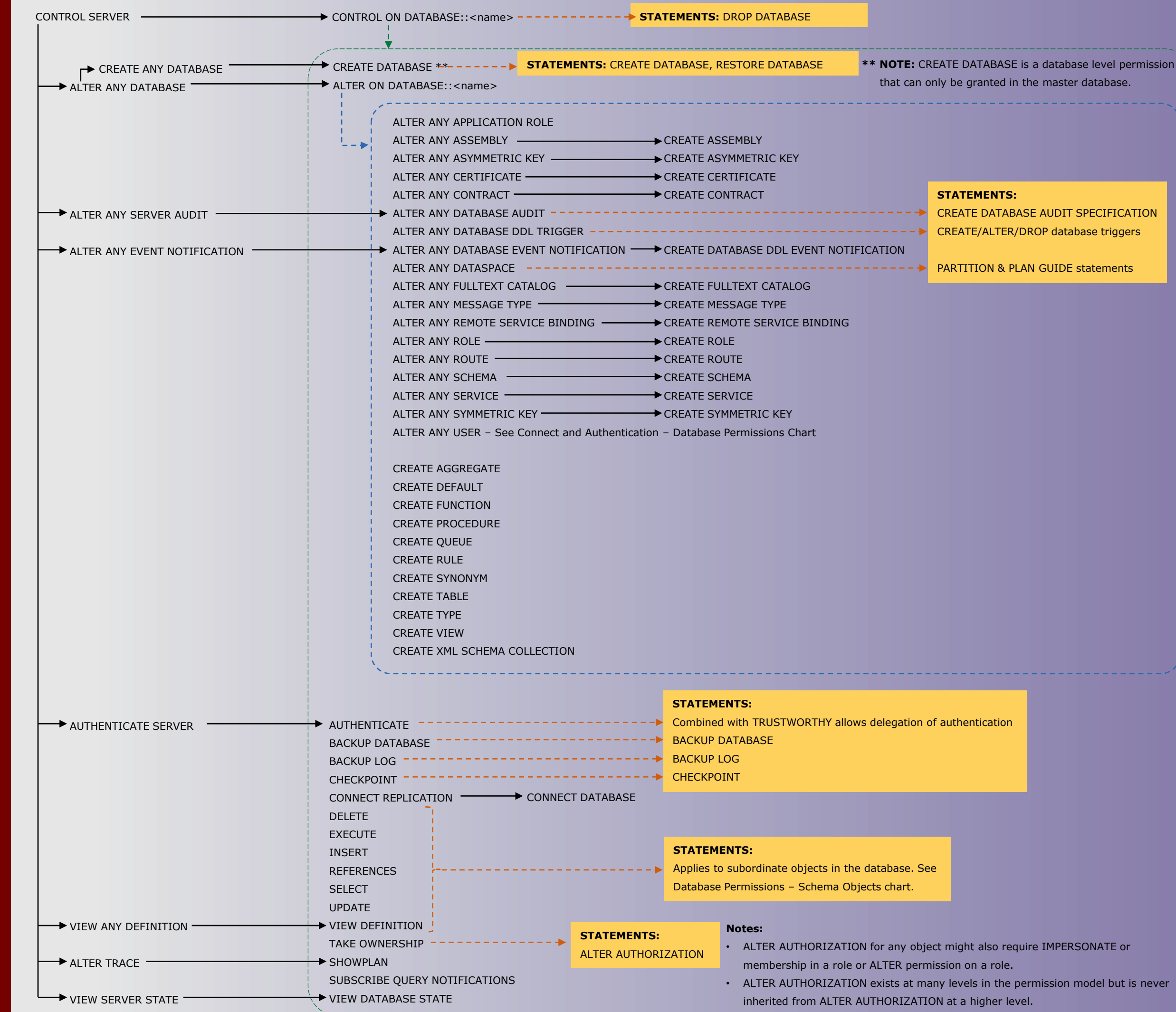
- CONTROL SERVER** sends a **CONTROL ON AVAILABILITY GROUP: <name>** command to the **CONTROL ON AVAILABILITY GROUP: <name>** component on the secondary server.
- CONTROL SERVER** sends a **VIEW ANY DEFINITION** command to the **VIEW DEFINITION ON AVAILABILITY GROUP: <name>** component on the secondary server.
- CONTROL SERVER** sends an **ALTER ANY AVAILABILITY GROUP** command to the **ALTER ON AVAILABILITY GROUP: <name>** component on the secondary server.
- CONTROL SERVER** sends a **CREATE AVAILABILITY GROUP** command to the **CREATE AVAILABILITY GROUP** component on the secondary server.

The components on the secondary server are grouped into a dashed box labeled **STATEMENTS:**

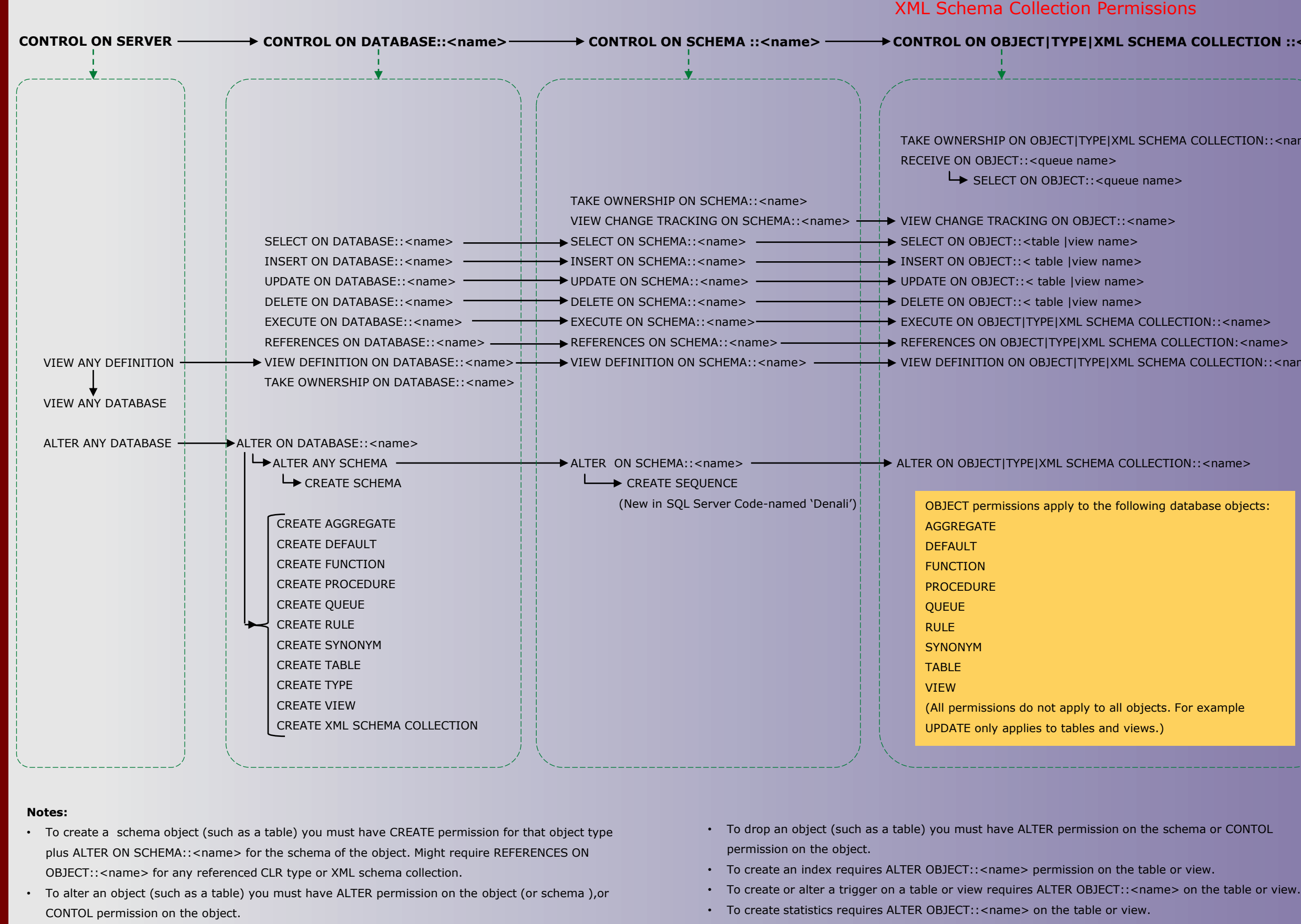
- VIEW DEFINITION ON AVAILABILITY GROUP: <name>**
- TAKE OWNERSHIP ON AVAILABILITY GROUP: <name>**
- ALTER ON AVAILABILITY GROUP: <name>**
- ALTER AVAILABILITY GROUP**
- DROP AVAILABILITY GROUP**
- CREATE AVAILABILITY GROUP**



Top Level Database Permissions



Server Permissions



Full-text search

CONTROL SERVER

CONTROL ON DATABASE::<name>

CONTROL ON FULLTEXT STOPLIST::<name>

CONTROL ON SEARCH PROPERTY LIST::<name>
(New in SQL Server Code-named 'Denali')

VIEW ANY DEFINITION

VIEW DEFINITION ON DATABASE::<name>

VIEW DEFINITION ON FULLTEXT CATALOG::<name>

VIEW DEFINITION ON FULLTEXT STOPLIST::<name>

VIEW DEFINITION ON SEARCH PROPERTY LIST::<name>
(New in SQL Server Code-named 'Denali')

REFERENCES ON DATABASE::<name>

REFERENCES ON FULLTEXT CATALOG::<name>

REFERENCES ON FULLTEXT STOPLIST::<name>

REFERENCES ON SEARCH PROPERTY LIST::<name>
(New in SQL Server Code-named 'Denali')

ALTER ANY DATABASE

ALTER ON DATABASE::<name>

ALTER ON FULLTEXT CATALOG::<name>

ALTER ON FULLTEXT STOPLIST::<name>

ALTER ON SEARCH PROPERTY LIST::<name>
(New in SQL Server Code-named 'Denali')

TAKE OWNERSHIP ON FULLTEXT CATALOG::<name>

TAKE OWNERSHIP ON FULLTEXT STOPLIST::<name>

TAKE OWNERSHIP ON SEARCH PROPERTY LIST::<name>
(New in SQL Server Code-named 'Denali')

CREATE FULLTEXT CATALOG

ALTER ANY FULLTEXT CATALOG

STATEMENTS:

- ALTER FULLTEXT CATALOG
- CREATE FULLTEXT CATALOG

STATEMENTS:

- ALTER FULLTEXT STOPLIST
- CREATE FULLTEXT STOPLIST

STATEMENTS:

- ALTER SEARCH PROPERTY LIST
- CREATE SEARCH PROPERTY LIST

NOTES:

- Creating a full-text index requires ALTER permission on the table and REFERENCES permission on the full-text catalog.
- Dropping a full-text index requires ALTER permission on the table.

```

graph LR
    CS[CONTROL SERVER] --> CDB[CONTROL ON DATABASE::<name>]
    CS --> CU[CONTROL ON USER::<name>]
    CDB -- "VIEW ANY DEFINITION" --> VDB[VIEW DEFINITION ON DATABASE::<name>]
    VDB --> VU[VIEW DEFINITION ON USER::<name>]
    CU --> IUS[IMPERSONATE ON USER::<name>]
    IUS --> IUS_BOX[STATEMENTS: EXECUTE AS]
    AD[ALTER ANY DATABASE] --> ADB[ALTER ON DATABASE::<name>]
    AU[ALTER ANY USER] --> AUA[ALTER ON USER::<name>]
    AUA --> AUA_BOX[STATEMENTS: ALTER USER, DROP USER, CREATE USER]
    COB[CONNECT ON DATABASE::<name>] --> COB_BOX[CONNECT ON DATABASE::<name>]
    AUA -.-> COB_BOX
  
```

NOTES:

- When contained databases are enabled, creating a database user that authenticates at the database, grants **CONNECT DATABASE** to that user, and it can access SQL Server without a login.
- Granting **ALTER ANY USER** allows a principal to create a user based on a login, but does not grant the server level permission to view information about logins.

CONTROL SERVER → **CONTROL ON DATABASE: <name>** → **CONTROL ON ROLE: <name>**

VIEW DEFINITION ON DATABASE: <name> → **VIEW DEFINITION ON ROLE: <name>**

ALTER ANY DATABASE → **ALTER ON DATABASE: <name>** → **TAKE OWNERSHIP ON ROLE: <name>**

ALTER ANY ROLE → **ALTER ON ROLE: <name>**

CREATE ROLE

STATEMENTS:

- ALTER ROLE <name> ADD MEMBER
- DROP ROLE
- CREATE ROLE

NOTES:

- Only members of the db_owner fixed database role can add or remove members from fixed database roles.
- Add members with sp_addmember in SQL Server 2008 R2. Use ALTER ROLE ... ADD MEMBER in SQL Server Code-named 'Denali'.

```

graph LR
    CS[CONTROL SERVER] --> CD[CONTROL ON DATABASE: <name>]
    CS --> CA[CONTROL ON APPLICATION ROLE: <name>]
    CS --> VD[VIEW ANY DEFINITION]
    CS --> AD[ALTER ANY DATABASE]
    CD --> VDD[VIEW DEFINITION ON DATABASE: <name>]
    CD --> VDR[VIEW DEFINITION ON APPLICATION ROLE: <name>]
    VD --> VDD
    VD --> VDR
    AD --> ADD[ALTER ON DATABASE: <name>]
    AD --> AAR[ALTER ANY APPLICATION ROLE]
    ADD --> AAR
    AAR --> AAR_AL[ALTER ON APPLICATION ROLE: <name>]
    AAR_AL --> ST[STATEMENTS:  
ALTER APPLICATION ROLE  
DROP APPLICATION ROLE  
CREATE APPLICATION ROLE]
  
```

The diagram illustrates the control flow from a **CONTROL SERVER** to a **DATABASE** and an **APPLICATION ROLE**. The **CONTROL SERVER** initiates control on both the database and the application role, as well as view definitions and database alterations. The **DATABASE** and **APPLICATION ROLE** components handle view definitions and alterations, with the application role component also listing specific statements like **ALTER APPLICATION ROLE**, **DROP APPLICATION ROLE**, and **CREATE APPLICATION ROLE**.

[illegible]

```

graph LR
    subgraph Left_Side [ ]
        direction TB
        CS[CONTROL SERVER]
        VAD[VIEW ANY DEFINITION]
        AAD[ALTER ANY DATABASE]
        CAK[CREATE ASYMMETRIC KEY]
    end

    subgraph Right_Side [ ]
        direction TB
        COK[CONTROL ON ASYMMETRIC KEY]
        VADOK[VIEW DEFINITION ON ASYMMETRIC KEY]
        ROK[REFERENCES ON ASYMMETRIC KEY]
        TOWOK[TAKE OWNERSHIP ON ASYMMETRIC KEY]
        AOK[ALTER ON ASYMMETRIC KEY]
    end

    CS --> COK
    CS --> VADOK
    CS --> ROK
    CS --> TOWOK
    CS --> AOK
    CS --> CAK

    VAD --> VADOK
    AAD --> ROK
    AAD --> TOWOK
    AAD --> AOK
    AAD --> CAK

    CAK --> AOK
  
```

Note: ADD SIGNATURE requires CONTROL permission on the key, and requires ALTER permission on the object.

STANETESTS:

- ALTER ASYMMETRIC KEY
- DROP ASYMMETRIC KEY
- CREATE ASYMMETRIC KEY

```

graph TD
    CS[CONTROL SERVER] --> COS[CONTROL ON DATABASE::<name>]
    AEAN[ALTER ANY EVENT NOTIFICATION] --> AEDN[ALTER ANY DATABASE EVENT NOTIFICATION]
    CDEN[CREATE DDL EVENT NOTIFICATION] --> CDDEN[CREATE DATABASE DDL EVENT NOTIFICATION]
    CTEEN[CREATE TRACE EVENT NOTIFICATION] --> CDTEN[CREATE DATABASE TRACE EVENT NOTIFICATION]
    COS -.-> AEDN
    AEDN -.-> CDDEN
    AEDN -.-> CDTEN
    CDDEN -.-> DBDDN[Database scoped DDL event notifications]
    CDTEN -.-> DBTEN[Event notifications on trace events]
  
```

Note: EVENT NOTIFICATION permissions also affect service broker. See the service broker chart for more info.

```
graph TD
    CS[CONTROL SERVER] --> CDB[CONTROL ON DATABASE::<name>]
    CDB --> CRD[CONNECT REPLICATION ON DATABASE::<name>]
    CRD --> COB[CONNECT ON DATABASE::<name>]
```

```

graph TD
    CS[CONTROL SERVER] -- "CONTROL ON DATABASE: <name>" --> DB[CONTROL ON DATABASE: <name>]
    DB -- "CONTROL ON CERTIFICATE: <name>" --> C[CONTROL ON CERTIFICATE: <name>]
    C -- "VIEW DEFINITION ON CERTIFICATE: <name>" --> DB
    C -- "REFERENCES ON CERTIFICATE: <name>" --> DB
    C -- "TAKE OWNERSHIP ON CERTIFICATE: <name>" --> DB
    DB -- "VIEW ANY CERTIFICATE" --> CS
    DB -- "ALTER ANY CERTIFICATE" --> CS
    DB -- "CREATE CERTIFICATE" --> CS
    
```

NOTE: ADD SIGNATURE requires control permission on the certificate, and requires ALTER permission on the object.

STATEMENTS:

- ALTER CERTIFICATE
- DROP CERTIFICATE
- CREATE CERTIFICATE

```

graph LR
    CS[CONTROL SERVER] -->|CONTROL ON DATABASE: <name>| DB[DATABASE]
    CS -->|CONTROL ON ASSEMBLY: <name>| AS[ASSEMBLY]
    CS -->|VIEW ANY DEFINITION| DB
    CS -->|ALTER ANY DATABASE| DB
    DB -->|VIEW DEFINITION ON DATABASE: <name>| AS
    DB -->|REFERENCES ON DATABASE: <name>| AS
    DB -->|ALTER ON DATABASE: <name>| AS
    AS -->|VIEW DEFINITION ON ASSEMBLY: <name>| DB
    AS -->|REFERENCES ON ASSEMBLY: <name>| DB
    AS -->|TAKE OWNERSHIP ON ASSEMBLY: <name>| DB
    AS -->|ALTER ON ASSEMBLY: <name>| DB
    AS -->|CREATE ASSEMBLY| DB
  
```

NOTE: CREATE and ALTER ASSEMBLY statements require server level EXTERNAL ACCESS ASSEMBLY and INVOKE ASSEMBLY permissions, and can require membership in the sysadmin fixed server role.

The diagram illustrates the mapping of SQL statements to database actions for three database objects: Database, Service, and Contract. Each object has a corresponding box with its actions and a yellow box with its statements. Arrows show the mapping from SQL statements to actions and then to the object boxes.

Database Object:

- CONTROL SERVER → CONTROL ON DATABASE: <name>
- VIEW ANY DEFINITION → VIEW DEFINITION ON DATABASE: <name>
- ALTER ANY DATABASE → ALTER ON DATABASE: <name>
- ALTER ANY SERVICE → ALTER ON SERVICE: <name>
- CREATE SERVICE → CREATE SERVICE

Service Object:

- CONTROL SERVER → CONTROL ON SERVICE BINDING: <name>
- VIEW ANY DEFINITION → VIEW DEFINITION ON REMOTE SERVICE BINDING: <name>
- ALTER ANY DATABASE → ALTER ON DATABASE: <name>
- ALTER ANY REMOTE SERVICE BINDING → ALTER ON REMOTE SERVICE BINDING: <name>
- CREATE REMOTE SERVICE BINDING → CREATE REMOTE SERVICE BINDING

Contract Object:

- CONTROL SERVER → CONTROL ON CONTRACT: <name>
- VIEW ANY DEFINITION → VIEW DEFINITION ON CONTRACT: <name>
- ALTER ANY DATABASE → ALTER ON DATABASE: <name>
- ALTER ANY CONTRACT → ALTER ON CONTRACT: <name>
- CREATE CONTRACT → CREATE CONTRACT

Statements (Yellow Boxes):

- Database:** ALTER SERVICE, DROP SERVICE, CREATE SERVICE
- Service:** ALTER REMOTE SERVICE BINDING, DROP REMOTE SERVICE BINDING, CREATE REMOTE SERVICE BINDING
- Contract:** DROP CONTRACT, CREATE CONTRACT

The diagram illustrates the relationship between database and route definitions for a view. It features four main components in rounded rectangles:

- VIEW DEFINITION ON DATABASE: <name>**: Contains the text "VIEW ANY DEFINITION" and "ALTER ANY DATABASE".
- VIEW DEFINITION ON ROUTE: <name>**: Contains the text "VIEW ANY ROUTE" and "ALTER ANY ROUTE".
- ALTER ON DATABASE: <name>**: Contains the text "ALTER ANY DATABASE".
- ALTER ON ROUTE: <name>**: Contains the text "ALTER ANY ROUTE".

Arrows indicate dependencies and updates:

- Green arrows point from the top to each of the four boxes.
- A solid black arrow points from "VIEW ANY DEFINITION" to "VIEW ANY ROUTE".
- A solid black arrow points from "ALTER ANY DATABASE" to "ALTER ANY ROUTE".
- A solid black arrow points from "VIEW ANY ROUTE" to "ALTER ON ROUTE".
- A solid black arrow points from "ALTER ANY ROUTE" to "ALTER ON ROUTE".
- A solid black arrow points from "ALTER ON ROUTE" to "ALTER ON DATABASE".
- A dashed orange arrow points from "ALTER ON DATABASE" to "ALTER ON ROUTE".
- A dashed orange arrow points from "CREATE ROUTE" to "ALTER ON ROUTE".

A yellow box on the right lists the statements:

- STATEMENTS:
- ALTER ROUTE
- DROP ROUTE
- CREATE ROUTE

The diagram illustrates the mapping of SQL commands to their corresponding database and message type actions. It is organized into two main columns: Database Actions and Message Type Actions.

Database Actions (Left Column):

- VIEW ANY DEFINITION
- ALTER ANY DATABASE
- ALTER ANY MESSAGE TYPE
- CREATE MESSAGE TYPE
- CREATE QUEUE

Message Type Actions (Right Column):

- VIEW DEFINITION ON MESSAGE TYPE::<name>
- REFERENCES ON MESSAGE TYPE::<name>
- TAKE OWNERSHIP ON MESSAGE TYPE::<name>
- ALTER ON MESSAGE TYPE::<name>

Statements (Bottom Right):

- ALTER MESSAGE TYPE
- DROP MESSAGE TYPE
- CREATE MESSAGE TYPE

Flow and Connections:

- VIEW ANY DEFINITION maps to VIEW DEFINITION ON MESSAGE TYPE::<name>.
- ALTER ANY DATABASE maps to ALTER ON DATABASE::<name>.
- ALTER ANY MESSAGE TYPE maps to ALTER ON MESSAGE TYPE::<name>.
- CREATE MESSAGE TYPE maps to CREATE MESSAGE TYPE.
- CREATE QUEUE maps to CREATE MESSAGE TYPE.
- ALTER ON MESSAGE TYPE::<name> maps to ALTER MESSAGE TYPE, DROP MESSAGE TYPE, and CREATE MESSAGE TYPE.

Notes:

- 1. The user executing the CREATE CATALOG statement must have REFERENCES permission on all message types specified.
- 2. The user executing the CREATE SERVICE statement must have REFERENCES permission on the queue and all contracts specified.
- 3. To execute the CREATE or ALTER REMOTE SERVICE BINDING the user must have impersonate permission for the principal specified in the statement.
- 4. When the CREATE or ALTER MESSAGE TYPE statement specifies a schema collection, the user executing the statement must have REFERENCES permission on the schema collection specified.
- 5. The ALTER ANY EVENT NOTIFICATION chart for more permissions related to Service Broker.
- 6. See the [SCHEMA OBJECTS chart](#) for [QUEL](#) permissions.
- 7. The SQL CATALOG statement exists but at this time there is no ALTER CATALOG statement.