TÖL303G

Gagnasafnsfræði

Snorri Agnarsson

BCNF þáttun

$R(A, B, C, D, E, F, G, H, I, J)$
Lykill: ABD. Þáttum á AB

Fallákveður:
$AB \rightarrow C$
$BD \rightarrow EF$
$AD \rightarrow GH$
$A \rightarrow I$
$H \rightarrow J$

$R_1(A, B, C, I)$
Lykill: AB. Þáttum á A

$R_2(A, B, D, E, F, G, H, J)$
Lykill: ABD. Þáttum á BD

$R_{11}(A, I)$
Lykill: A. BCNF

$R_{12}(A, B, C)$
Lykill: AB. BCNF

$R_{21}(B, D, E, F)$
Lykill: BD. BCNF

$R_{22}(A, B, D, G, H, J)$
Lykill: ABD. Þáttum á AD

$R_{221}(A, D, G, H, J)$
Lykill: AD. Þáttum á H

$R_{222}(A, B, D)$
Lykill: ABD. BCNF

$R_{2211}(H, J)$
Lykill: H. BCNF

$R_{2212}(A, D, G, H)$
Lykill: AD. BCNF

BCNF þáttun 2

Fallákveður:

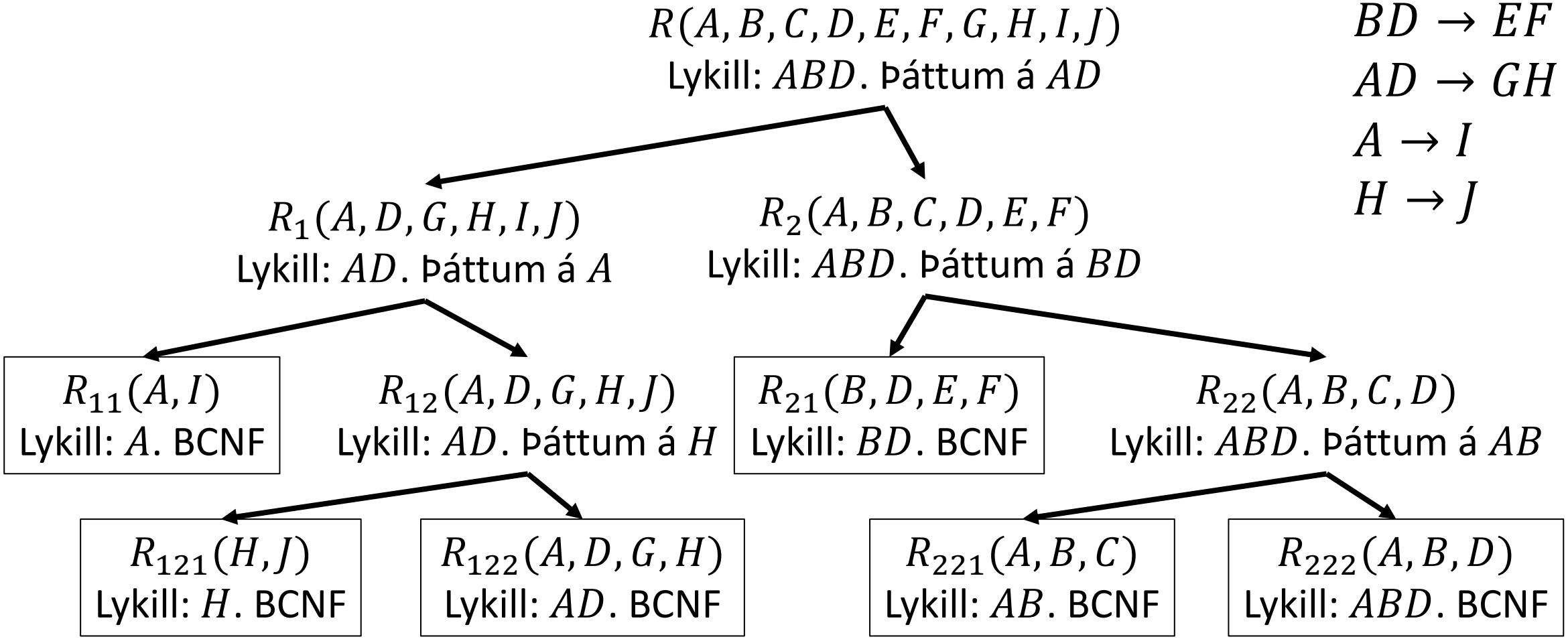
$AB \rightarrow C$

$BD \rightarrow EF$

$AD \rightarrow GH$

$A \rightarrow I$

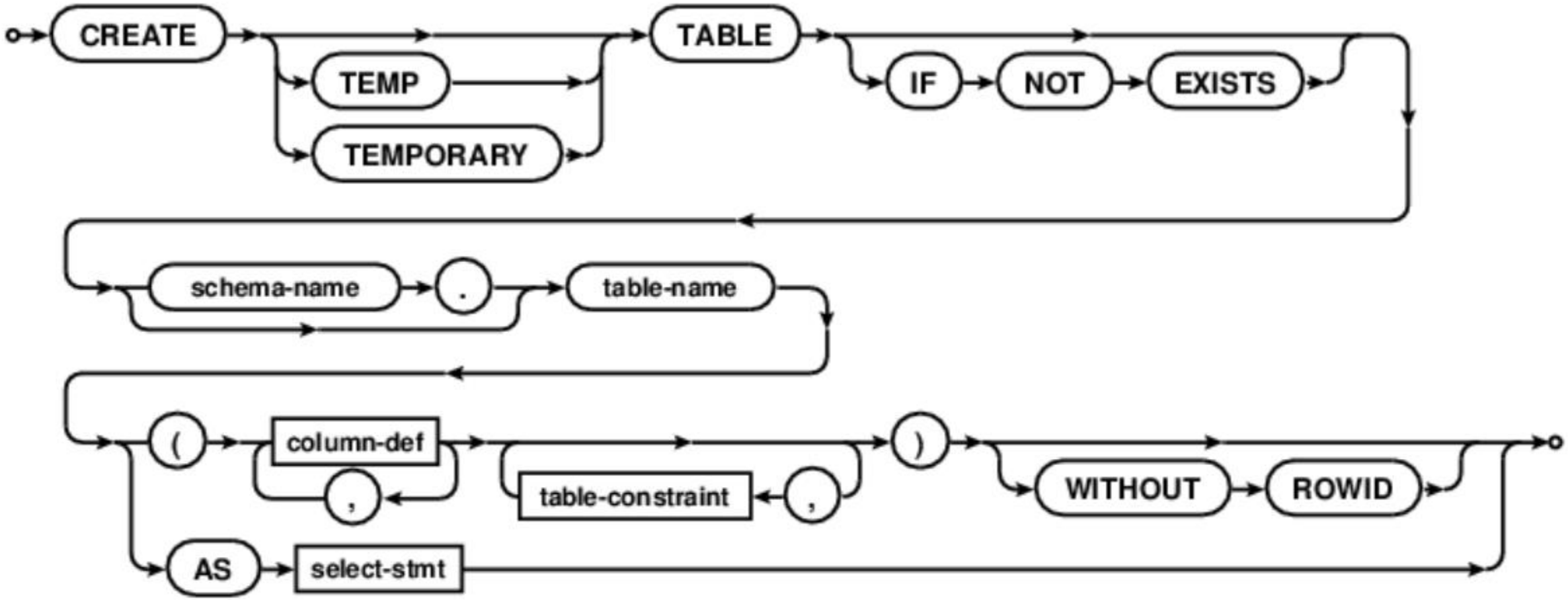
$H \rightarrow J$



Skorður í SQL – Constraints in SQL

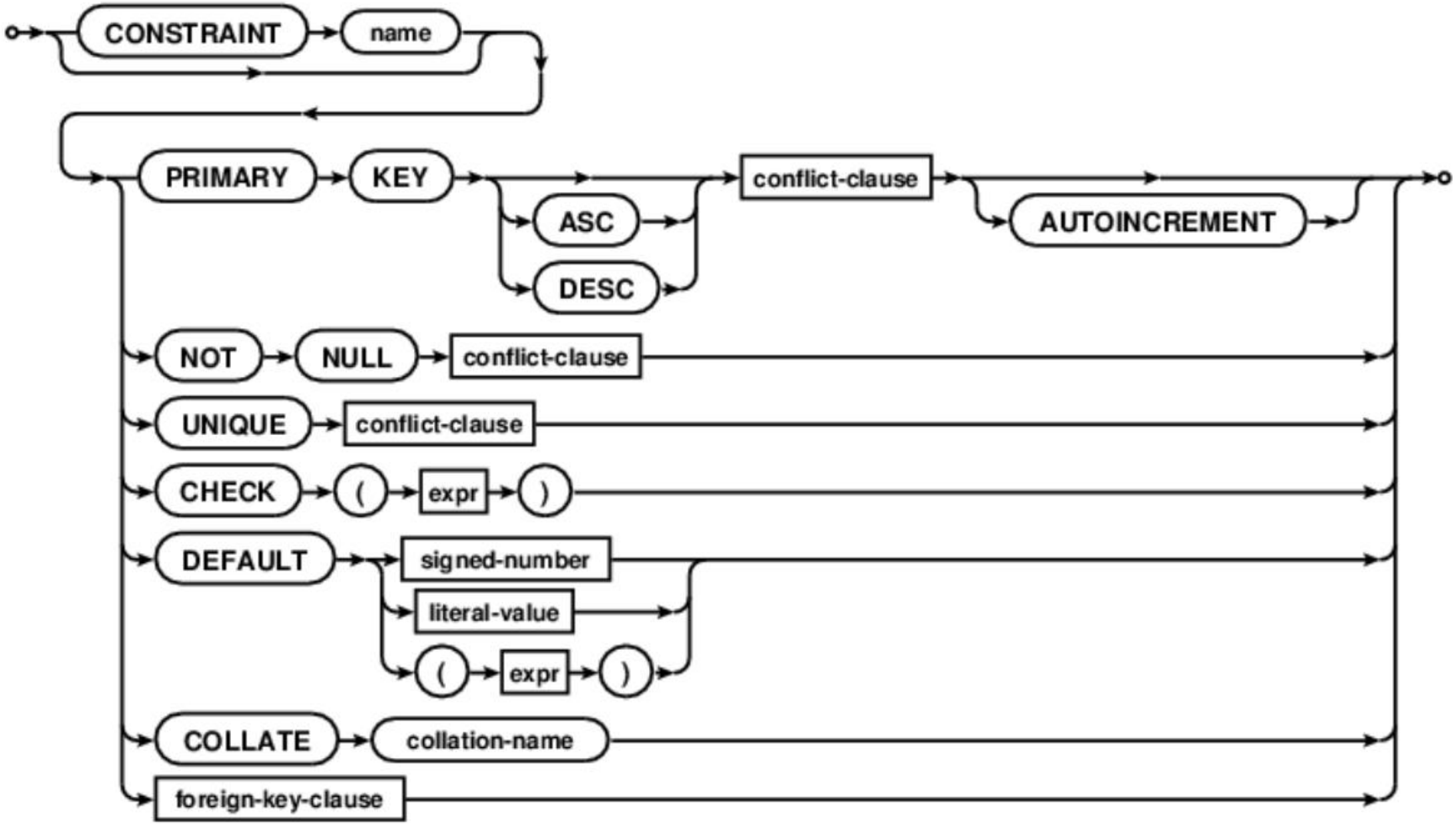
- Algengustu SQL skorðurnar eru heilleikaskorður (referential integrity)
  - Heilleikaskorður tryggja að tengingar milli tafla vísi ekki út í bláinn
  - Referential integrity means that references between tables refer to valid rows
- FOREIGN KEY skilgreinir heilleikaskorður/defines referential constraints
- PRIMARY KEY skilgreinir aðallykil/defines primary key
  - Einkennir röð – Distinguishes rows
  - Má ekki innihalda NULL – Must not contain NULL
  - Aðeins einn aðallykill er leyfilegur – Only one primary key is allowed
- UNIQUE tilgreinir annan lykil – UNIQUE defines another key
  - Einkennir einnig röð – Also distinguishes row
  - Megum hafa marga slíka – May have multiple such
  - Má innihalda NULL (aðeins eitt samt) – May contain NULL (but only one)
- NOT NULL skorða bannar NULL gildi í viðkomandi dálki
  A NOT NULL constraint forbids NULL values in the column

create-table-stmt: hide



column-def: hide

column-constraint: hide



foreign-key-clause

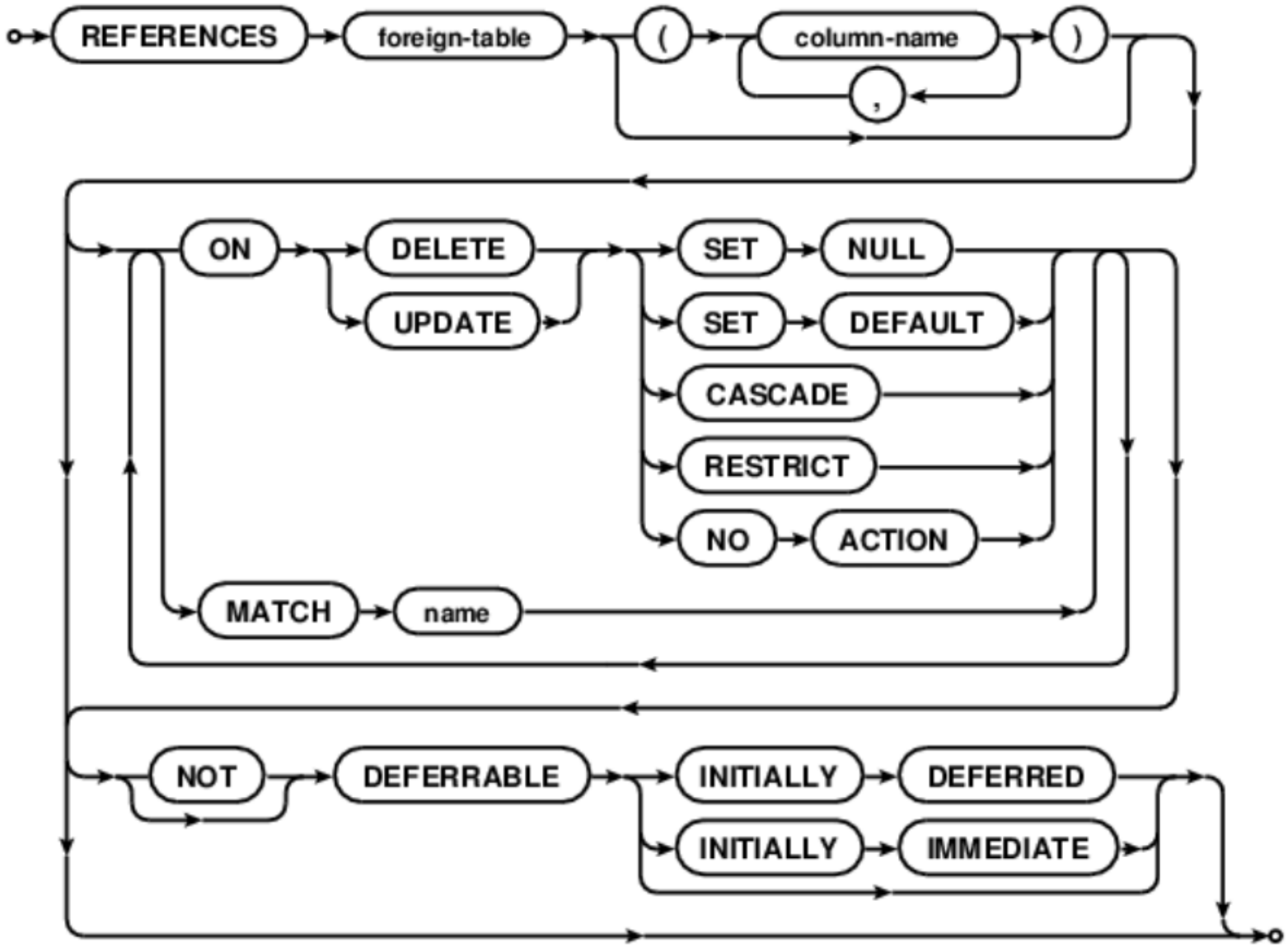
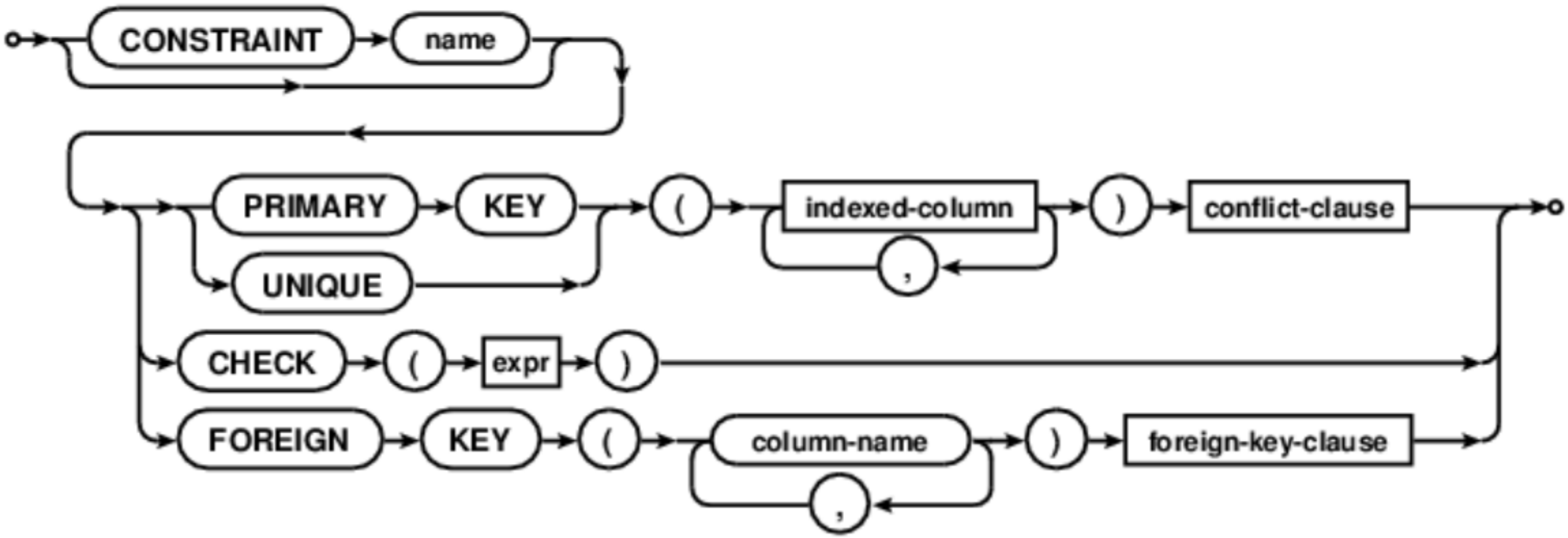
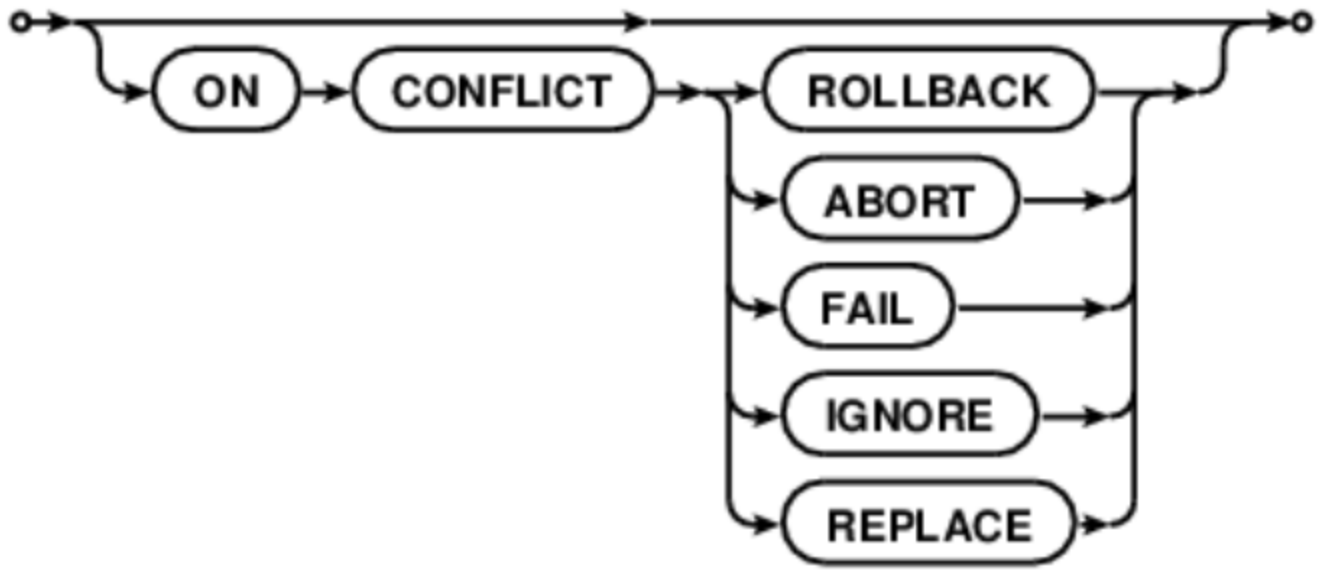


table-constraint



conflict-clause



NOT NULL

- Í sumum tilvikum viljum við ekki leyfa NULL gildi, til dæmis fyrir price í tölvugagnagrunninum – We may want to forbid NULL values for the price column in the computers database
- Þetta er gert með því að skilgreina NOT NULL fyrir viðkomandi dálk

```
CREATE TABLE T
        ( A VARCHAR(32)
        , B INT NOT NULL
        , ID INT PRIMARY KEY
        , C INT UNIQUE
        );
```

FOREIGN KEY

- Hingað til hafa tengingar milli tafla verið óformlegar
  Hitherto references between tables have been informal
- Dálkarnir (title,year) tengja saman Movie og StarsIn töflurnar
  The columns (title,year) join the Movie and StarsIn tables
  - En það er ekki tryggt að tengingin sé til staðar
    But it is not ensured that the reference works

```
CREATE TABLE Movie
        ( title VARCHAR(25)
        , year INT
        , length INT
        , inColor INT
        , studioName VARCHAR(15)
        , producerC VARCHAR(3)
        );
```

```
CREATE TABLE StarsIn
        ( movieTitle VARCHAR(25)
        , movieYear INT
        , starName VARCHAR(30)
        );
```

PRIMARY KEY

- Fyrst setjum við (title,year) sem aðallykil (primary key)
  First we specify (title,year) as the primary key

```
CREATE TABLE Movie
        ( title VARCHAR(25)
        , year INT
        , length INT
        , inColor INT
        , studioName VARCHAR(15)
        , producerC VARCHAR(3)
        , PRIMARY KEY(title,year)
        );
```

- Til að tryggja að (title,year) sé ávallt lykill í Movie
  To ensure that (title,year) is always a key in Movie

FOREIGN KEY

```
CREATE TABLE Movie
        ( title VARCHAR(25)
        , year INT
        , length INT
        , inColor INT
        , studioName VARCHAR(15)
        , producerC VARCHAR(3)
        , PRIMARY KEY (title,year)
        );
```

```
CREATE TABLE StarsIn
        ( movieTitle VARCHAR(25)
        , movieYear INT
        , starName VARCHAR(30)
                REFERENCES
                        MovieStar(name)
        , FOREIGN KEY
                (movieTitle,movieYear)
                REFERENCES
                        Movie(title,year)
);
```

- Í SQLite þurfum við einnig að framkvæma
  In SQLite we also need to execute
  PRAGMA FOREIGN_KEYS=ON;

Tilgangur – Purpose

- Með því að skilgreina aðallykil (Primary Key) tryggjum við að viðkomandi dálkar myndi lykil og að aðeins ein röð innihaldi hvern lykil
  - Það er ekki hægt að setja inn fleiri en eina kvikmynd í Movie töfluna með sama nafni og ártali – We ensure that each movie has a unique (title,year)
- Með því að skilgreina ytri lykil (Foreign Key) tryggjum við að til sé samsvarandi röð í foreldristöflunni (þeirri töflu sem vísað er í)
  - Það er ekki hægt að setja inn röð í StarsIn nema viðkomandi kvikmynd sé til í Movie töflunni – The Foreign Key constraint makes it impossible to insert a row into StarsIn unless the movie exists in the Movie table
- Með skorðunum erum við að tilgreina hluta af **fastayrðingu gagna** fyrir gagnagrunnninn – The constraints become parts of the **data invariant** for the database

Kvikmyndagagnagrunnurinn

- Lögum núna gagnagrunninn
- Tökum afrit af gagnagrunninum með .dump skipun

```
sqlite3 f8.db .dump > f8.sql
```

- Fáum gagnagrunninn á textasniði, skilgreiningar á töflum ásamt gögnum sem runa SQL skipana

Hvenær? When?

- Skorður eru skilgreindar
  - Með skemanu sem hluti af CREATE TABLE ...
    - Yfirleitt staðfest í lok fjöldainnsetningar
  - Eða seinna, með ALTER TABLE ...
    - Takmörkuð virkni í SQLite, en hægt að gera allt með smá tilfæringum svo sem endurnefningum tafla
    - Staðfest á þeim gögnum sem eru til staðar
- Í hvert skipti sem við breytum gögnum eru skorður staðfestar
  - INSERT: Ný gildi þurfa að uppfylla skorður
  - UPDATE: Sama
  - DELETE: Ytri lyklar annars staðar verða ennþá að vísa á lykla í töflunni
- Getum seinkað staðfestingu skorða með því að nota (í REFERENCES klausu)

  DEFERRABLE INITIALLY DEFERRED

When?

- Constraints are defined
  - With the schema as part of CREATE TABLE ...
    - Usually validated at the end of a transaction
  - Or later, with ALTER TABLE ...
    - Limited capability in SQLite, but possible using tricks such as renaming tables
    - Validated on the existing data
- Each time we make changes the constraints are validated
  - INSERT: New values must fulfil contraints
  - UPDATE: Ditto
  - DELETE: Foreign keys in other tables must still refer to existing keys in the table
- Can defer constraint validation by using (in REFERENCES clause)
  DEFERRABLE INITIALLY DEFERRED

Röð innsetninga – Order of insertions

- Ef við keyrum – If we run:

```
PRAGMA foreign_keys=on;
INSERT INTO StarsIn(movieTitle,movieYear,starName)
      VALUES('Glengarry Glen Ross',1992,'Alec Baldwin');
```

- í nýja gagnagrunninum f8b.db þá kvartar SQLite – SQLite complains for f8b.db
  Hins vegar virkar þetta – However this works:

```
PRAGMA foreign_keys=on;
INSERT INTO Movie(title,year, length, inColor, StudioName,
      producerC) VALUES ('Glengarry Glen Ross', 1992,
      100,1,'GGR',333);
INSERT INTO StarsIn(movieTitle,movieYear,starName)
      VALUES('Glengarry Glen Ross',1992,'Alec Baldwin');
```

Ytri lyklar og breytingar – Foreign keys and changes

- Ef við höfum ytri lykil í S.B (dálkur B í töflu S) sem vísar í R.A (lykill A í töflu R), hvaða breytingar geta brotið skilyrðið?

  If we have a foreign key in S.B (column B in table S) that refers to R.A (the key A in table R), what changes can break the constraint?

  - INSERT inn í S – INSERT into S
  - UPDATE á S.B – UPDATE on S.B
  - DELETE úr R – DELETE from R
  - UPDATE á R.A – UPDATE on R.A

Breytingar? Changes?

- Hvernig eigum við að meðhöndla brot á skorðum?
  How should we handle breaks of constraints?
- INSERT inn í S – INSERT on S
  - Villa – Error
- UPDATE á S.B – UPDATE on S.B
  - Villa – Error
- DELETE úr R (DELETE from R) eða/or UPDATE á R.A (UPDATE on R.A)
  - Villa? Error?
  - Setja NULL í (fyrrum samsvarandi) S.B?
    Put NULL into (previously corresponding) S.B?
  - Eyða (fyrrum samsvarandi) röðum úr S?
    Delete (previously corresponding) rows from S?

Viðbrögð við brotnum tilvísunum

- Þegar gildi í dálki sem vísað er á í ytri lykli er breytt eru þrír valkostir:
  1. Skila villu – þetta er sjálfgefin hegðun, valin með RESTRICT lykilorði
  2. Setja (fyrrum) tilvísun sem NULL, valin með SET NULL
  3. Breyta eða eyða (fyrrum) tilvísun, eftir atvikum, valið með CASCADE
- Hegðun fyrir DELETE og UPDATE má velja óháð hvoru öðru

```
CREATE TABLE Movie
        ( ...
        , producerC VARCHAR(3) REFERENCES MovieExec(cert)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        );
```

- Ef við breytum MovieExec.cert þá breytast samsvarandi Movie.producerC gildi
- Ef við eyðum MovieExec röð þá fá (fyrrum) samsvarandi Movie.producerC gildið NULL

Responding to broken constraints

- When a value is changed in a column that is referred to by a foreign key there are three options:
  1. Cause an exception – This is the default, also chosen with the RESTRICT keyword
  2. Make the (previous) reference NULL, chosen with SET NULL
  3. Change or delete the (previous) reference, as the case may be, chosen with CASCADE
- Responses for DELETE and UPDATE may be independently chosen

```
CREATE TABLE Movie
        ( ...
        , producerC VARCHAR(3) REFERENCES MovieExec(cert)
                ON DELETE SET NULL
                ON UPDATE CASCADE
        );
```

- If we change MovieExec.cert then the corresponding Movie.producerC values change correspondingly
- If we delete a row from MovieExec then the (previously) corresponding Movie.producerC values change to NULL

Jaðartilvik – An extreme case

```
CREATE TABLE T
     ( A INT, B INT, C INT
     , PRIMARY KEY(A,B)
     , FOREIGN KEY (B,C)
          REFERENCES T(A,B)
          ON DELETE CASCADE
     );
```

A	B	C
1	1	1
2	1	1
3	2	1
4	3	2
5	4	3
6	5	4
7	6	5
8	7	6

- Hvað gerist ef við framkvæmum? – What happens if we execute?

```
DELETE FROM T WHERE A=1;
```

Jaðartilvik – An extreme case

```
CREATE TABLE T
     ( A INT, B INT, C INT
     , PRIMARY KEY(A,B)
     , FOREIGN KEY (B,C)
          REFERENCES T(A,B)
          ON DELETE CASCADE
     );
```

A	B	C
1	1	1
2	1	1
3	2	1
4	3	2
5	4	3
6	5	4
7	6	5
8	7	6

- Hvað gerist ef við framkvæmum
  `DELETE FROM T WHERE A=1;`

ÖLL GILDI HVERFA ÚR TÖFLUNNI
ALL VALUES DISAPPEAR

Almennar skorður – General constraints

- Við getum sett skorður á gildi í dálki með CHECK
  We can constrain the values in a column with CHECK

```
CREATE TABLE MovieExec
      ( name VARCHAR(30)
      , address VARCHAR(30)
      , cert VARCHAR(3) PRIMARY KEY
      , netWorth INT CHECK(netWorth>1000000)
      );
```

- Sum gagnagrunnskerfi leyfa hlutfyrirspurnir í CHECK, en ekki SQLite
  Some DBMS‘s allow nested queries in CHECK, but not SQLite

Almennar skorður – General constraints

- Við getum blandað dálkum í skorðum
  We can mix columns in constraints

```
CREATE TABLE MovieStar
        ( name VARCHAR(30) PRIMARY KEY
        , address VARCHAR(30)
        , gender CHAR(1) CHECK(gender in (‘F‘, ‘M‘))
        , birthdate VARCHAR(10)
        , CONSTRAINT RightTitle
                CHECK(gender=‘F‘ OR name NOT LIKE ‘Ms.%‘)
        );
```

Fastayrðingar – Data invariants

- SQL skilgreinir fastayrðingar (assertion) sem eru mjög öflugar en eru því miður ekki studdar í flestum kerfum
  SQL defines data invariants (assertions) which are very powerful but are sadly not supported in most systems

```
CREATE ASSERTION NoColorMoviesWithPoorMovieExecs
        CHECK(
                NOT EXISTS(
                        SELECT * FROM Movie,MovieExec
                        WHERE
                                cert=producerC AND
                                inColor=1 AND
                                netWorth<1000000
                )
        );
```

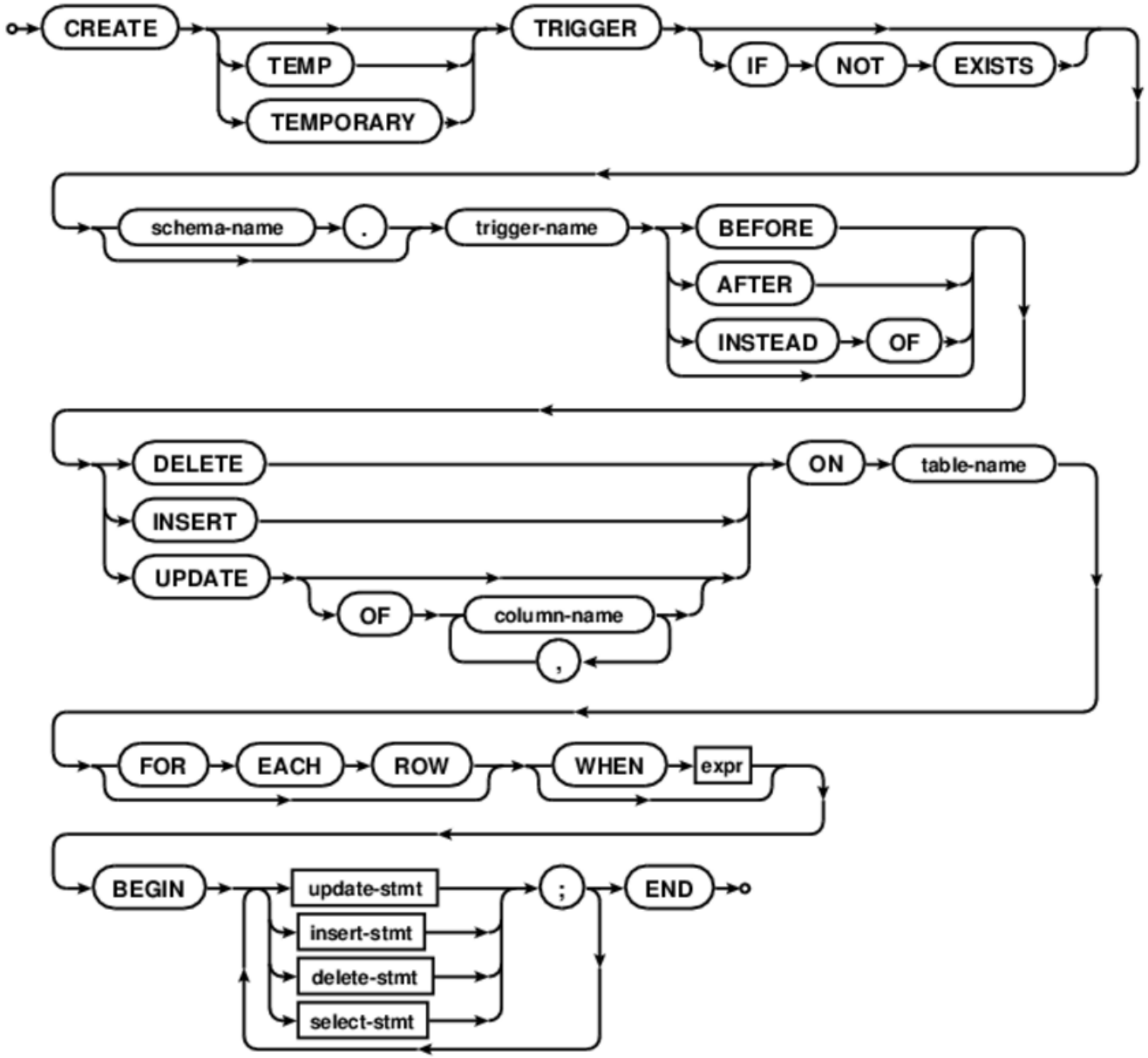
Gikkir – Triggers

- Skorður takmarka innihald gagnagrunns
  Constraints limit the possible contents of a database
- Gikkir geta aðlagað gögnin að skorðunum
  Triggers can conform the data to the constraints

```
CREATE TRIGGER NetWorthTrigger
AFTER UPDATE OF netWorth ON MovieExec
FOR EACH ROW
WHEN( OLD.netWorth > NEW.netWorth)
BEGIN
        UPDATE MovieExec
        SET netWorth = OLD.netWorth
        WHERE cert=NEW.cert;
END;
```

- Í SQLite eru gikkir til í einfaldri útgáfu – SQLite has a limited form of triggers
  - Fyrir UPDATE er vísað til nýju og gömlu raðanna með OLD og NEW
    For UPDATE we refer to the new and old rows with OLD and NEW
  - Fyrir INSERT er aðeins NEW til staðar
    For INSERT only NEW is available
  - Fyrir DELETE er aðeins OLD til staðar
    For DELETE only OLD is available

create-trigger-stmt: hide



SQL forritun

ODBC, JDBC, ESQL, ...

Yfirlit – Overview

- SQL inni í öðrum forritunarmálum
  SQL inside other programming languages
  - Innfellt (embedded) SQL (ESQL), kvikt (dynamic) SQL, SQLJ
    Embedded SQL (ESQL), dynamic SQL, SQLJ
- Gagnagrunnsforritun gegnum staðlaðar einingar:
  SQL/CLI (t.d. ODBC) og JDBC
  Database programming through standardized modules: SQL/CLI (e.g. ODBC) and JDBC
- **Geymd stef** (föll) í gagnagrunnum
  Stored procedures (functions) in databases
- Samanburður á aðferðum – Comparison of methods