Dispensing with DDL

An account from having tried.

DDL – origins

- Multiple languages common even in earliest DBMS systems
  - IDMS
    - Data Manipulation Language
    - Schema Definition Language
    - Device Media Control Language
    - Distinct compilers
    - Access to compiler main means for securing access to the language ???

DDL – origins

- Multiple languages common even in earliest DBMS systems
  - SQL
    - Data Manipulation Language
    - Data Definition Language
    - Compiler identity is unexposed implementation detail
    - Security still main reason for distinct language ???

Codd's 4th rule

- DBMS systems required to have a catalog

Codd's 4th rule

- DBMS systems required to have a catalog
- That is accessible to users using their regular query language

Codd's 4th rule

- DBMS systems required to have a catalog
- That is accessible to users using their regular query language
- Accessibility limited to “interrogation” ?

Cause and consequence

- DDL = cause, DML = consequence
  - CREATE TABLE …
  - INSERT INTO SYSTEM.SYSTABLES ...
- Same DML not allowed as 'cause'

Consequences

- Extra language/grammar to master
  - More demanding on the user's (mental) ability

Consequences

- Extra language/grammar to master
  - More demanding on the user's (mental) ability
- Extra parsers/compilers to be built
  - More occasion to make errors, more testing work to be done, more maintenance work to be done, ...

Alternatives possible ?

- There must be a catalog,

Alternatives possible ?

- There must be a catalog,
- Which is essentially “just another database”

Alternatives possible ?

- There must be a catalog,
- Which is essentially “just another database”
- Which must therefore be updatable, even if only “indirectly”

Alternatives possible ?

- There must be a catalog,
- Which is essentially “just another database”
- Which must therefore be updatable, even if only “indirectly”
- ===> Catalog DML equivalent to DDL

Alternatives possible ?

- DML already includes :
  - Set-level updates
    - CREATE TABLE X ..., TABLE Y... ???

Alternatives possible ?

- DML already includes :
  - Set-level updates
  - Mixed delete+insert (= assignment)
    - ALTER TABLE … ADD column1, DROP column2 ???

Alternatives possible ?

- DML already includes :
  - Set-level updates
  - Mixed delete+insert (= assignment)
  - Multiple assignment
    - ALTER TABLE … ADD column1,
      ADD KEY … { … , column1} ???

Alternatives possible ?

- DML already includes :
  - Set-level updates
  - Mixed delete+insert (= assignment)
  - Multiple assignment
  - D_INSERT vs. INSERT
    - CREATE TABLE IF NOT EXISTS vs. ...

Alternatives possible ?

- DML already includes :
  - Set-level updates
  - Mixed delete+insert (= assignment)
  - Multiple assignment
  - D_INSERT vs. INSERT
- ===> CDML strictly more powerful than DDL

Standard language ?

- CREATE TABLE <tablename> vs. INSERT INTO SYSRELVARS <relation>
  - CREATE corresponds to INSERT
  - TABLE corresponds to SYSRELVARS
  - <tablename> corresponds to inserted content in <relation>

Standard language ?

- CREATE TABLE <tablename> vs. INSERT INTO SYSRELVARS <relation>
  - CREATE corresponds to INSERT
  - TABLE corresponds to SYSRELVARS
  - <tablename> corresponds to inserted content in <relation>
- But TABLE is a keyword of the language,

Standard language ?

- CREATE TABLE <tablename> vs. INSERT INTO SYSRELVARS <relation>
  - CREATE corresponds to INSERT
  - TABLE corresponds to SYSRELVARS
  - <tablename> corresponds to inserted content in <relation>
- But TABLE is a keyword of the language,
- SYSRELVARS is a relvar name in a database

Standard language ?

- Keywords of the language can be part of a standard,

Standard language ?

- Keywords of the language can be part of a standard,
- Can catalog relvar names be too ?

Standard language ?

- Keywords of the language can be part of a standard,
- Can catalog relvar names be too ?
- Cfr. INFORMATION_SCHEMA

Consequences for the user ?

- CDML more resemblant to the DML that the user will know already.

Consequences for the user ?

- CDML more resemblant to the DML that the user will know already.
- CDML likely has less keywords
- [because many keywords replaced with relvar names]

Consequences for the user ?

- CDML more resemblant to the DML that the user will know already.
- CDML likely has less keywords
- [because many keywords replaced with relvar names]
- But user will be familiar with those relvar names already, otherwise how could he ever interrogate the catalog ?

Processing CDML

- Must be “hooked onto” database updates as they get carried out.
  - INSERT INTO SYSKEY … , INSERT INTO SYSKEYATTRIBUTE … ;
    - ===> verify current database data
    - ===> refresh internal runtime objects representing the constraint
  - INSERT INTO SYSFILE … ;
    - ===> physically create and perhaps format the file in the OS filesystem

Processing CDML

- {UpdateType Relvar} 1-1 to “system action”
  - UpdateType = {INSERT|DELETE}
  - “System action” =
    AddKeyAction, DeleteIndexAction, …
  - Some actions might be No-op's.

Processing CDML

Getting the action :

```
class UpdateLuw extends Luw {
    …
    void doDeletionsAndInsertions (Statement statement) {
        for (Map<Relvar,Relation> me : statement.getInserts().entrySet()) {
            iRelvar = me.getKey();
            for (Tuple it : me.getValue()) {
                actions = iRelvar.getDeferredSystemActionForInsertion(it)
            }
        }
    }
}

class BaseRelvar extends Relvar {
    final Set<DeferredSystemAction> getDeferredSystemActionForInsertion (Tuple t) {

        return deferredSystemActionsFactory != null ?
                deferredSystemActionsFactory.getDeferredSystemAction(t,
OperationType.INSERT)
               : new HashSet<DeferredSystemAction>();

}
```

Processing CDML

Relvar constructors :

```
class BaseRelvar extends Relvar {
    BaseRelvar (…) {
        this.deferredSystemActionsFactory = … ;
    }
}

class UserRelvar extends BaseRelvar {
    UserRelvar (…) {
        super(…, null) ;  /* User relvars have no associated action factory */
    }
}

class SysRelvar extends BaseRelvar {
    SysRelvar (…) {
        super(…, SystemDefinedActionFactories.get(…)) ;
                 /* Sys relvars have predefined action factories */
    }
}
```

Processing CDML

- Examples of actions sets for a single relvar :
  - {AddKey AddRelvar AddKeyAttribute}
  - {AddKey AddKeyAttribute}

Processing CDML

- Examples of actions sets for a single relvar :
  - {AddKey AddRelvar AddKeyAttribute}
  - {AddKey AddKeyAttribute}
- Actions are different !
  - Corresponding key constraint not checked
  - Corresponding key constraint checked

Processing CDML

- Examples of actions sets for a single relvar :
  - {AddKey AddRelvar AddKeyAttribute}
  - {AddKey AddKeyAttribute}
- Actions are different !
  - Corresponding key constraint not checked
  - Corresponding key constraint checked
- Some actions not to be executed because of presence of other actions

Processing CDML

- Examples of actions sets for a relvar :
  - {AddAttribute AddKey AddKeyAttribute AddRecordAttribute}

Processing CDML

- Examples of actions sets for a relvar :
  - {AddAttribute AddKey AddKeyAttribute AddRecordAttribute}
- Multiple actions :
  - Value for new attribute to be recorded in physical records
  - Corresponding key constraint checked

Processing CDML

- Examples of actions sets for a relvar :
  - {AddAttribute AddKey AddKeyAttribute AddRecordAttribute}
- Multiple actions :
  - Value for new attribute to be recorded in physical records
  - Corresponding key constraint checked
- Some actions to be executed in a specific order relative to other actions

Processing CDML

- Getting the order (plus skippable actions) :
  - For each possible pair of actions :
  - PrecedenceRegulator Class
  - Via Class.forName based on action names

Processing CDML

- Getting the order (plus skippable actions) :

```
class UpdateLuw extends Luw {
    …
    void doDeletionsAndInsertions (Statement statement) {
        …
        DeferredSystemActionPreferenceRegulator preferenceRegulator = (DeferredSystem-
ActionPreferenceRegulator) Class.forName(preferenceRegulatorClassName).newInstance();

        preferenceResult = preferenceRegulator.getPreferenceBetween(otherAction, inser-
tionDeferredSystemAction, this);
    }
}

final class AddKeyAddRelvarPreferenceRegulator extends DeferredSystemActionPreferen-
ceRegulator {
PreferenceResult getPreferenceBetweenIdentifiedActionTypes (DeferredSystemAction ac-
tion1, DeferredSystemAction action2, UpdateLuw luw) {
    AddKeyAction addKeyAction = (AddKeyAction) action1;
    AddRelvarAction addRelvarAction = (AddRelvarAction) action2;
    if (addKeyAction.getRelvarName().equals(addRelvarAction.getRelvarName())) {
        return PreferenceResult.REPLACEDBY;
    } else {
        return PreferenceResult.NOPREFERENCE;
    }
}
```

Processing CDML

- Examples of actions sets for a relvar :
  - {AddRecordAttribute}
  - {AddRecordAttribute DeleteRecordAttribute}

Processing CDML

- Examples of actions sets for a relvar :
  - {AddRecordAttribute}
  - {AddRecordAttribute DeleteRecordAttribute}
- Actions are once again different :
  - Extend existing records
  - Desirable to do a rearrange-in-place

Processing CDML

- Examples of actions sets for a relvar :
  - {AddRecordAttribute}
  - {AddRecordAttribute DeleteRecordAttribute}
- Actions are once again different :
  - Extend existing records
  - Desirable to do a rearrange-in-place
- ===> Desirable to “detect updates”

Processing CDML

- Examples of actions sets for a physical file :
  - {DeleteFile DeleteStorageSpace}
  - {DeleteFile DeleteStorageSpace AddFile}

Processing CDML

- Examples of actions sets for a physical file :
  - {DeleteFile DeleteStorageSpace}
  - {DeleteFile DeleteStorageSpace AddFile}
- Actions are once again different :
  - {Delete existing file}
  - {Resize the pages of an existing file, delete some storage space within the file}

Processing CDML

- Examples of actions sets for a physical file :
  - {DeleteFile DeleteStorageSpace}
  - {DeleteFile DeleteStorageSpace AddFile}
- Actions are once again different
- DeleteStorageSpace can only be skipped in the first scenario !

Processing CDML

- Examples of actions sets for a physical file :
  - {DeleteFile DeleteStorageSpace}
  - {DeleteFile DeleteStorageSpace AddFile}
- Actions are once again different
- DeleteStorageSpace can only be skipped in the first scenario !
- Detecting updates is of higher priority than detecting action ordering and skipping

Processing CDML

Detecting updates :

```
class UpdateLuw extends Luw {
    …
    //step 2 : detect and filter out all the UPDATES
    for (DeferredSystemAction deferredSystemAction : unorderedDeferredSystemActions) {
        for (DeferredSystemAction otherDeferredSystemAction :
unorderedDeferredSystemActionsCopy) {
            deferredSystemAction.detectAndHandleUpdates(otherDeferredSystemAction);
            }
        }
    }
}

final class AddDbmsFileAction extends DeferredSystemAction {
    void detectAndHandleUpdates (DeferredSystemAction otherDeferredSystemAction) {
        if (otherDeferredSystemAction instanceof DeleteDbmsFileAction) {
            if (otherDeferredSystemAction.getFileName() == this.getFileName()) {
                this.update = true;
                otherDeferredSystemAction.skipExecution();
            }
        }
    }
}
```

Detecting updates, a digression

- Update = Delete-then-Insert

Detecting updates, a digression

- Update = Delete-then-Insert
- At least conceptually …

Detecting updates, a digression

- Update = Delete-then-Insert
- At least conceptually …
- Also at implementation level ?
  - Delete record {Erwin Smout, 07 NOV 1963}
  - Delete index entry “Erwin Smout”
  - Insert record {Erwin Smout, 07 NOV 1983}
  - Insert index entry “Erwin Smout”

Detecting updates, a digression

- Update = Delete-then-Insert
- At least conceptually …
- Also at implementation level ?
  - Update record {Erwin Smout, 07 NOV 1963} in-place ?

Detecting updates, a digression

- Update = Delete-then-Insert
- At least conceptually …
- Also at implementation level ?
- Detecting updates desirable for any DML !

Detecting updates, a digression

- Update = Delete-then-Insert
- Detecting updates desirable for any DML !
- But …which key ?
  - CDML System Actions : logical identifier of the database object

Detecting updates, a digression

- Update = Delete-then-Insert
- Detecting updates desirable for any DML !
- But …which key ?
  - CDML System Actions : logical identifier of the database object
  - General DML write performance : attributes used as the clustering key for the physical record location
  - Both not necessarily equal ...

Validating CDML

- DBMS has its own “business rules”, just like any other application ...

Validating CDML

- DBMS has its own “business rules”, just like any other application ...
- Old-style DDL : “application controls” coded in the DDL processor.

Validating CDML

- DBMS has its own “business rules”, just like any other application ...
- CDML-style : just another DB constraint on the catalog !
  - “Every relvar must have at least one key”

    ```
    ISEMPTY(RELVAR NOT MATCHING KEY)
    ```

  - “Physical records' lengths cannot exceed the available size of the pages they are written on”

    ```
    ISEMPTY (SUMMARIZE … JOIN … WHERE RECLEN > PAGELEN);
    ```

Validating CDML

- DBMS has its own “business rules”, just like any other application ...
- CDML-style : just another DB constraint on the catalog !
- Support for declarative DB constraints is sine qua non !

Validating CDML

- DBMS has its own “business rules”, just like any other application ...
- CDML-style : just another DB constraint on the catalog !
- Support for declarative DB constraints is sine qua non,
- But not sufficient !

Validating CDML

- Cases of CDML validation not supported by DB constraints on the catalog :
  - Upon defining a new DB constraint, current data must satisfy that constraint.
  - Upon defining/updating anything involving an expression (virtual relvar, constraints, ...), that expression must compile successfully against the post-update (!) catalog value.

Validating CDML

- Upon defining a new DB constraint, current data must satisfy that constraint :
  - Check is part of the AddDatabaseAction.
  - Ordering of the actions must be such that the query can indeed be successfully evaluated.
    - e.g. add new relvar attribute,
      with a default value,
      plus a tuple constraint on that attribute

Validating CDML

- DB objects involving expressions
  - e.g. delete attribute x from R1,
    add view V as “R1 WHERE x>0”;

Validating CDML

- DB objects involving expressions
  - Addressed by Assignment constraints,

Validating CDML

- DB objects involving expressions
  - Addressed by Assignment constraints,
  - That involve an “introspection” operator (EXPRESSIONINFO)

Validating CDML

- DB objects involving expressions
  - Addressed by Assignment constraints,
  - That involve an “introspection” operator (EXPRESSIONINFO),
  - Which takes the expression as a STRING argument,

Validating CDML

- DB objects involving expressions
  - Addressed by Assignment Constraints,
  - That involve an “introspection” operator (EXPRESSIONINFO),
  - Which takes the expression as a STRING argument,
  - And which addresses not the current, but the post-update value of the catalog
  - (which is why this rule cannot be defined as a DB constraint)

Mixing CDML with “regular” DML

- Consider D statement

```
VAR BASE RELATION {…} somename := RELATION { … }
```

Mixing CDML with “regular” DML

- Consider D statement

```
VAR BASE RELATION {…} somename := RELATION { … }
```

- Two actions in single statement
  - Declare relvar (=CDML-like operation)
  - Initialize relvar (=DML-like operation)

Mixing CDML with “regular” DML

- Consider D statement
  `VAR BASE RELATION {…} somename := RELATION { … }`
- Two actions in single statement
  - Declare relvar (=CDML-like operation)
  - Initialize relvar (=DML-like operation)
- = MA with “mixed” targets ?

Mixing CDML with “regular” DML

- Consider D statement

```
VAR BASE RELATION {…} somename := RELATION { … }
```

- Actions for the system to do :
  - Assign to catalog (to define the relvar)
  - Check catalog constraints
  - Assign to new user relvar
  - Check user DB constraints (because init value might be in violation of declared keys etc.)

Mixing CDML with “regular” DML

- Consider D statement

```
VAR BASE RELATION {…} somename := RELATION { … }
```

- <=> all checks performed before any update

Mixing CDML with “regular” DML

- Consider D statement
  ```
  VAR BASE RELATION {…} somename := RELATION { … }
  ```
- <=> all checks performed before any update
- Internal runtime objects will have been built to assign to the new relvar, and to evaluate expressions against it.

Mixing CDML with “regular” DML

- Consider D statement
  `VAR BASE RELATION {…} somename := RELATION { … }`
- <=> all checks performed before any update
- Internal runtime objects will have been built to assign to the new relvar, and to evaluate expressions against it.
- Internal runtime objects are thus not “final” after the catalog update portion is done !

Mixing CDML with “regular” DML

- Consider D statement

```
VAR BASE RELATION {…} somename := RELATION { … }
```

- Conclusion : currently outlawed in any form

CDML and locking

- Relvars cannot be manipulated while “under alteration”, be it wrt:
  - Its logical structure/definition
  - Its physical implementation
  - Its associated constraints
    - Total relvar constraint
    - Any assignment constraint for this relvar
  - The triggers associated with the relvar

CDML and locking

- Relvars cannot be manipulated while “under alteration”
- Solution with 2PL on the relvar as a lockable object :
  - All queries of and all updates to the relvar require a “shared” (“read”) lock.
  - All catalog updates affecting the relvar require an “exclusive” lock.

CDML and commit/rollback

- When to actually execute SystemActions ?
- At end-of-statement ?
  - But might involve work on unrecoverable resources such as OS files …
  - That might nonetheless still need to be rolled back later on …

CDML and commit/rollback

- When to actually execute SystemActions ?
- At commit time ?
  - But then in the “mean” time, the issuing transaction should already “see” the new logical structure of the relvar,
  - Which requires internal runtime objects,
  - Which might have to be discarded upon eventual rollback

CDML and commit/rollback

- When to actually execute SystemActions ?
- ===> either option implies rather serious complications on implementation level

CDML and commit/rollback

- When to actually execute SystemActions ?
- ===> either option implies rather serious complications on implementation level
- ===> designer choice : all CDML is always immediately committed, regardless of the containing transaction's autocommit setting.

CDML

- That's all, folks.