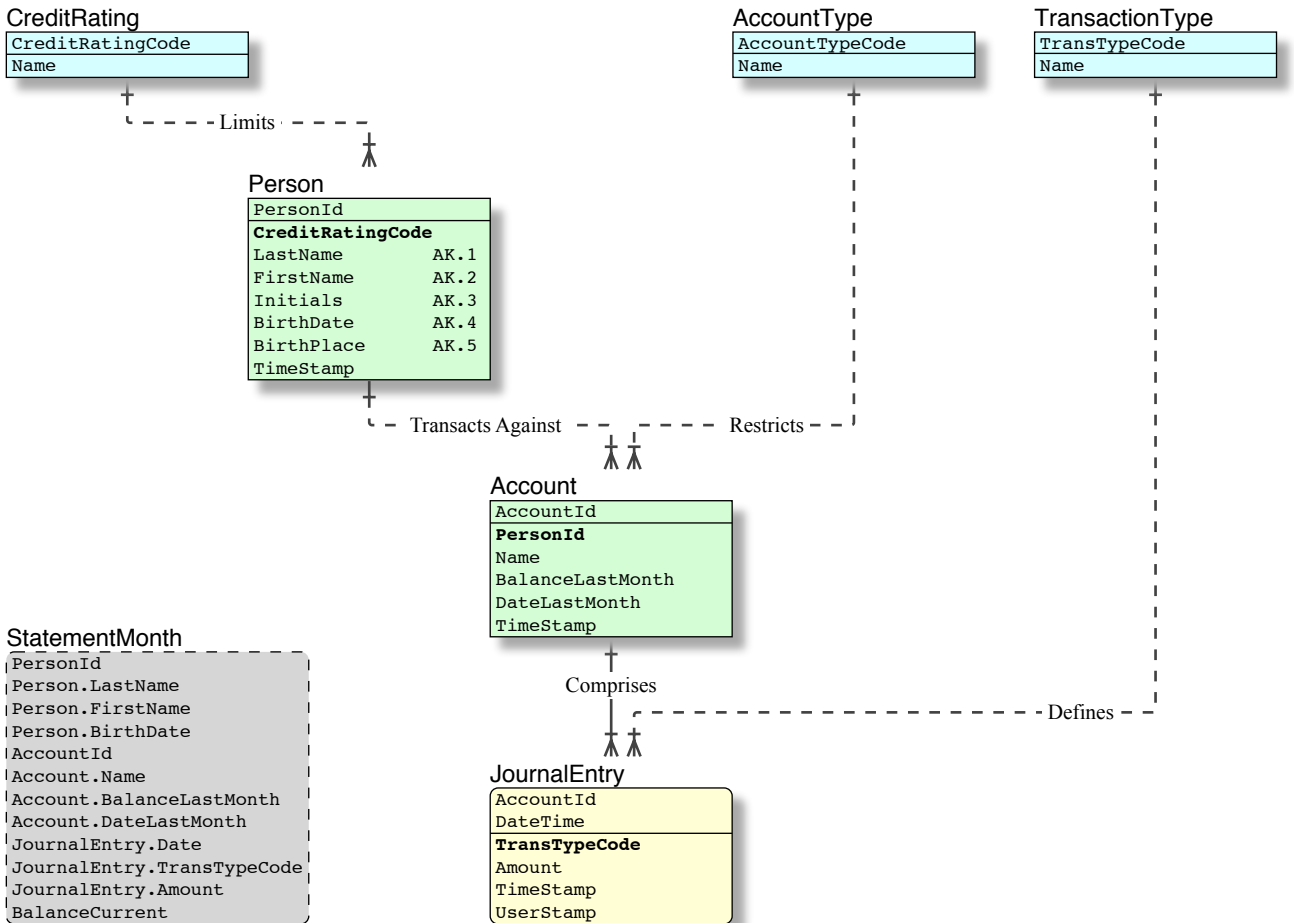
Transaction TTM 101122

Bank Account

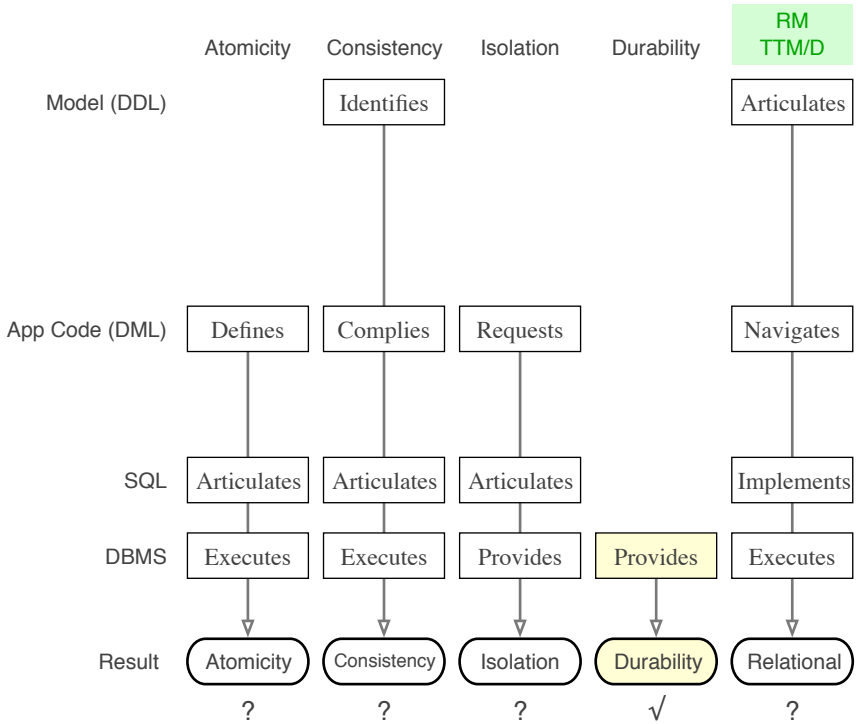


IDEF1X Notation

Transaction Grid

	Atomicity	Consistency	Isolation	Durability	RM TTM/D
Model (DDL)		Identifies			Articulates
App Code (DML)	Defines	Complies	Requests		Navigates
SQL	Articulates	Articulates	Articulates		Implements
DBMS	Executes	Executes	Provides	Provides	Executes

Transaction ACID ?



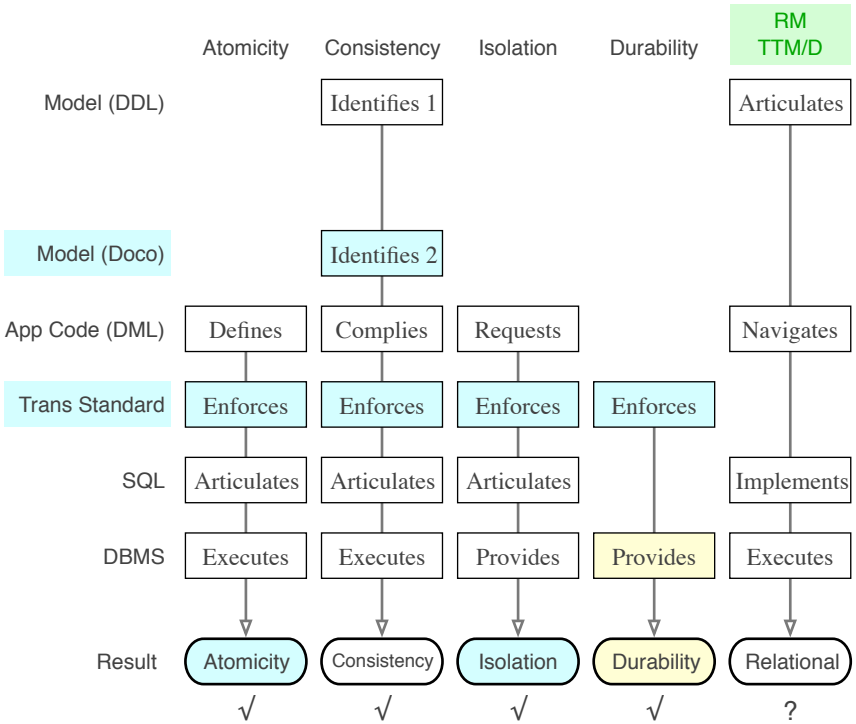
Optimistic Locking ?

Concurrency ?

Lost Updates ?

Hang on, er, what exactly, is a transaction ?

Transaction Standard



Optimistic Locking ?

Concurrency ?

Lost Updates ?

Lost Update

Time 1

Person P1 has a Credit Rating A which allows $10,000 overdraft
Person P1 fails to keep agreements re covering their overdraft
Bank Officer changes Person P1 Credit Rating D; $0 overdraft

```
SELECT  @CreditRatingCode = CreditRatingCode,
        @BirthPlace       = BirthPlace
    FROM Person
    WHERE PersonId = P1
```

User interaction, uncontrolled duration

Time 2

Person P1 has a balance of $5,000 DR (overdrawn)
Person P1 phones bank clerk
Checks that they still have Credit Rating A
Changes their address

```
SELECT  @CreditRatingCode = CreditRatingCode,
        @BirthPlace       = BirthPlace
    FROM Person
    WHERE PersonId = P1
```

User interaction, uncontrolled duration

Time 3

```
BEGIN TRAN ONE
UPDATE Person SET
    CreditRatingCode ="D",
    BirthPlace       = @BirthPlace
    WHERE PersonId = P1
COMMIT TRAN ONE
```

Time 4

```
BEGIN TRAN TWO
UPDATE Person SET
    CreditRatingCode = @CreditRatingCode,
    BirthPlace       = "Timbuktu"
    WHERE PersonId = P1
COMMIT TRAN TWO
```

Person P1 attempts to withdraw $2,000

Time 5

```
BEGIN TRAN Withdraw
SET TRANSACTION ISOLATION LEVEL 3
SELECT  @CreditRatingCode = CreditRatingCode,
        @Balance = BalanceLastMonth + (
        SELECT SUM(Amount)
            FROM JournalEntry
            WHERE AccountId = a.AccountId
            AND   DateTime >  a.DateLastMonth
            )
    FROM Person p,
         Account a
    WHERE p.PersonId  = @PersonId
    AND   p.PersonId  = a.PersonId
    AND   a.AccountId = @AccountId
SET @Allowed = CASE @CreditRatingCode
    WHEN "A" THEN @Balance + 10000
    WHEN "B" THEN @Balance + 1000
    WHEN "C" THEN @Balance + 100
    WHEN "D" THEN @Balance
    END
IF @Request <= @Allowed
    BEGIN
    INSERT JournalEntry VALUES (
        @AccountId,
        GETDATE(),
        "DR",
        @Request
        GETDATE(),
        SUSER_ID()
        )
    COMMIT TRAN WITHDRAW
    RETURN 0 -- Transaction Accepted
    END
ELSE
    BEGIN
    COMMIT TRAN WITHDRAW
    RETURN 1 -- Transaction Rejected
    END
```

Accepted due to Credit Rating A

```
-- The error checking, which is mandatory for every verb, is excluded for brevity
```

Optimistic Locking

Time 1

Person P1 has a Credit Rating A which allows $10,000 overdraft
Person P1 fails to keep agreements re covering their overdraft
Bank Officer changes Person P1 Credit Rating D; $0 overdraft

```
SELECT  @CreditRatingCode = CreditRatingCode,
        @BirthPlace       = BirthPlace,
        @TimeStamp        = TimeStamp
    FROM Person
    WHERE PersonId = P1
```

User interaction, uncontrolled duration

Time 2

Person P1 has a balance of $5,000 DR (overdrawn)
Person P1 phones bank clerk
Checks that they still have Credit Rating A
Changes their address

```
SELECT  @CreditRatingCode = CreditRatingCode,
        @BirthPlace       = BirthPlace,
        @TimeStamp        = TimeStamp
    FROM Person
    WHERE PersonId = P1
```

User interaction, uncontrolled duration

Time 3

```
BEGIN TRAN PersonUpdate
SET TRANSACTION ISOLATION LEVEL 3
SELECT  1
    FROM Person
    WHERE PersonId   = @PersonId
    AND   TimesStamp = @TimesStamp
IF @@ROWCOUNT != 1
    BEGIN
    ROLLBACK TRAN PersonUpdate
    RETURN 20003
    END
UPDATE Person SET
        CreditRatingCode = @CreditRatingCode,
        BirthPlace       = @BirthPlace
    WHERE PersonId   = @PersonId
    AND   TimesStamp = @TimesStamp
IF @@ROWCOUNT != 1
    BEGIN
    ROLLBACK TRAN PersonUpdate
    RETURN 20003
    END
COMMIT TRAN PersonUpdate
RETURN 0
```

Parameter

```
@PersonId,
@TimeStamp,
@CreditRatingCode = "D",
@BirthPlace = Birthplace[T1]
```

Return

```
@ReturnValue
```

Time 4

```
BEGIN TRAN PersonUpdate
SET TRANSACTION ISOLATION LEVEL 3
SELECT  1
    FROM Person
    WHERE PersonId   = @PersonId
    AND   TimesStamp = @TimesStamp
IF @@ROWCOUNT != 1
    BEGIN
    ROLLBACK TRAN PersonUpdate
    RETURN 20003
    END
UPDATE Person SET
        CreditRatingCode = @CreditRatingCode,
        BirthPlace       = @BirthPlace
    WHERE PersonId   = @PersonId
    AND   TimesStamp = @TimesStamp
IF @@ROWCOUNT != 1
    BEGIN
    ROLLBACK TRAN PersonUpdate
    RETURN 20003
    END
COMMIT TRAN PersonUpdate
RETURN 0
```

Parameter

```
@PersonId,
@TimeStamp,
@CreditRatingCode = CreditRatingCode[T2],
@BirthPlace = "Timbuktu"
```

Return

```
@ReturnValue
```

Fails Due To

```
TimesStamp[T3] != TimesStamp[T2]
```

Fresh Retrieval Eliminates Lost Update

Time 5

Person P1 attempts to withdraw $2,000
Rejected due to Credit Rating D

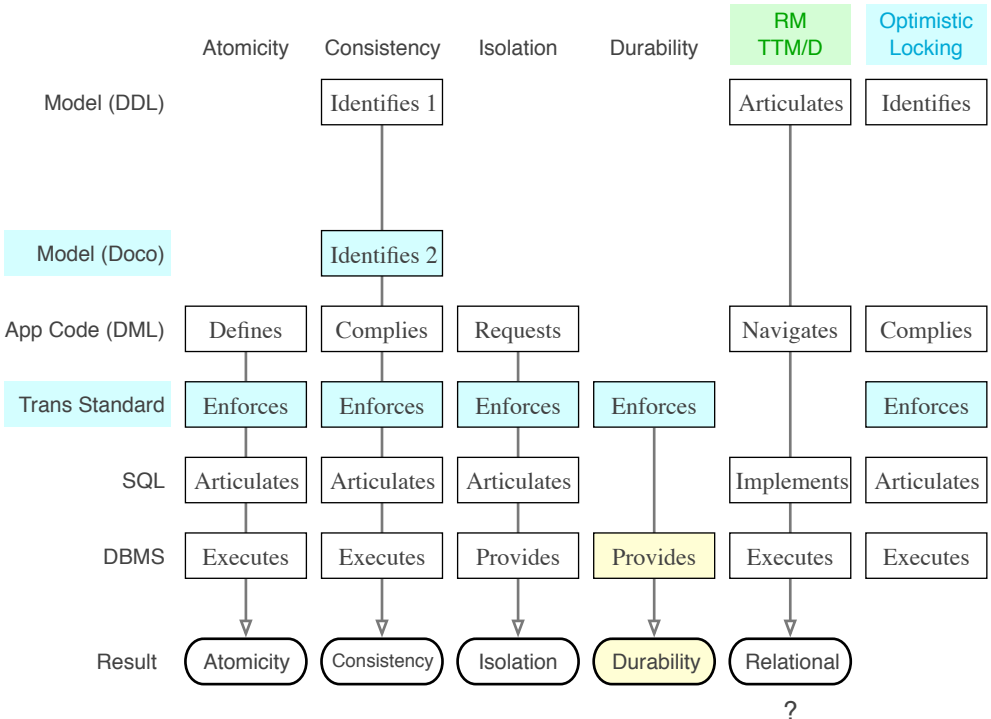
```
BEGIN TRAN Withdraw
SET TRANSACTION ISOLATION LEVEL 3
SELECT  @CreditRatingCode = CreditRatingCode,
        @Balance = BalanceLastMonth + (
        SELECT SUM(Amount)
            FROM JournalEntry
            WHERE AccountId = a.AccountId
            AND   DateTime >  a.DateLastMonth
            )
    FROM Person p,
         Account a
    WHERE p.PersonId   = @PersonId
    AND   p.TimesStamp = @TimesStamp
          -- any change re Person
    AND   p.PersonId   = a.PersonId
    AND   a.AccountId  = @AccountId
IF @@ROWCOUNT != 1
    BEGIN
    ROLLBACK TRAN PersonUpdate
    RETURN 20003
    END
...
```

Larger Transactions

This device can be further optimised; storing and checking every row within a multiple-row transaction may not be necessary. The `TimeStamp` of the top-most row can be used to identiy the status of the transaction tree. Here the `Person.TimeStamp` is used to indicate any change to the Person's `Person`, `Account` or `JournalEntry` rows.

```
-- The error checking, which is mandatory for every verb, is excluded for brevity
-- Error 20001: This transaction is Atomic, it cannot be executed from an open transaction; you have a transaction open.
-- Error 20002: This subtransaction must be executed from an open transaction; you do not have a transaction open.
-- Error 20003: The database has changed between retrieval and transaction execution; retrieve your data again.
```

Transaction/Optimistic Locking



√ Optimistic Locking

√ Concurrency

√ No Lost Updates