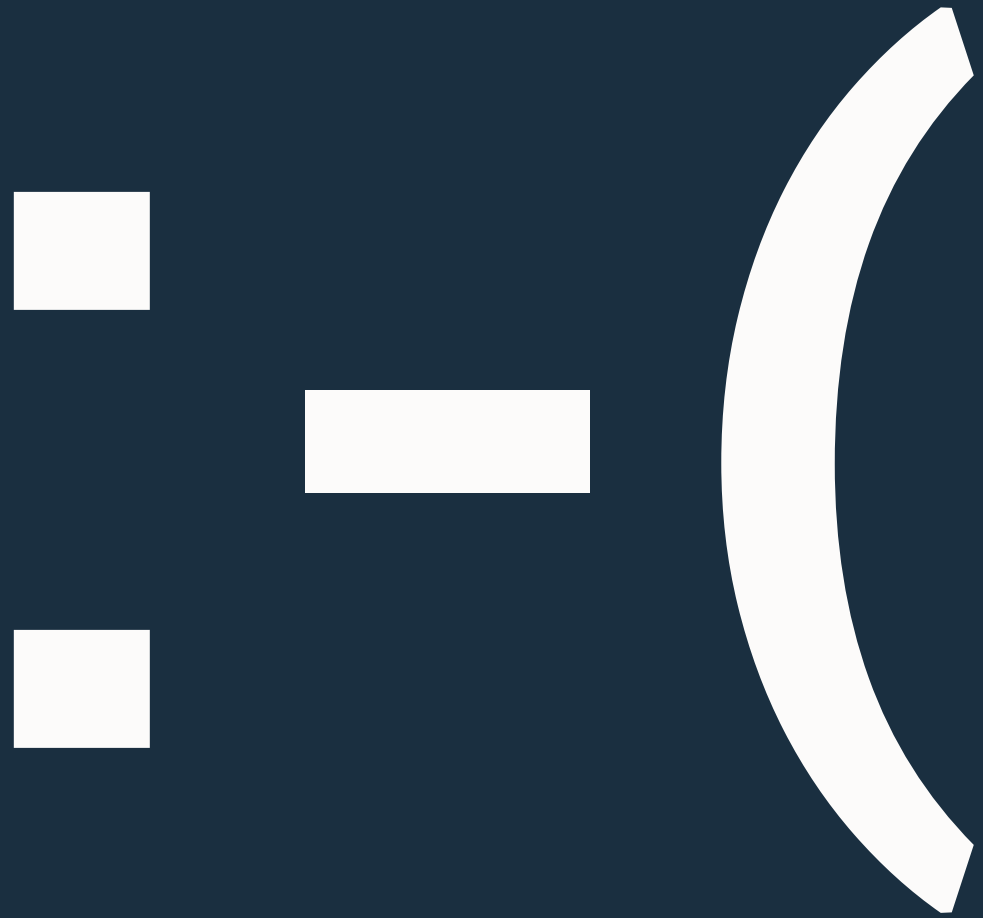
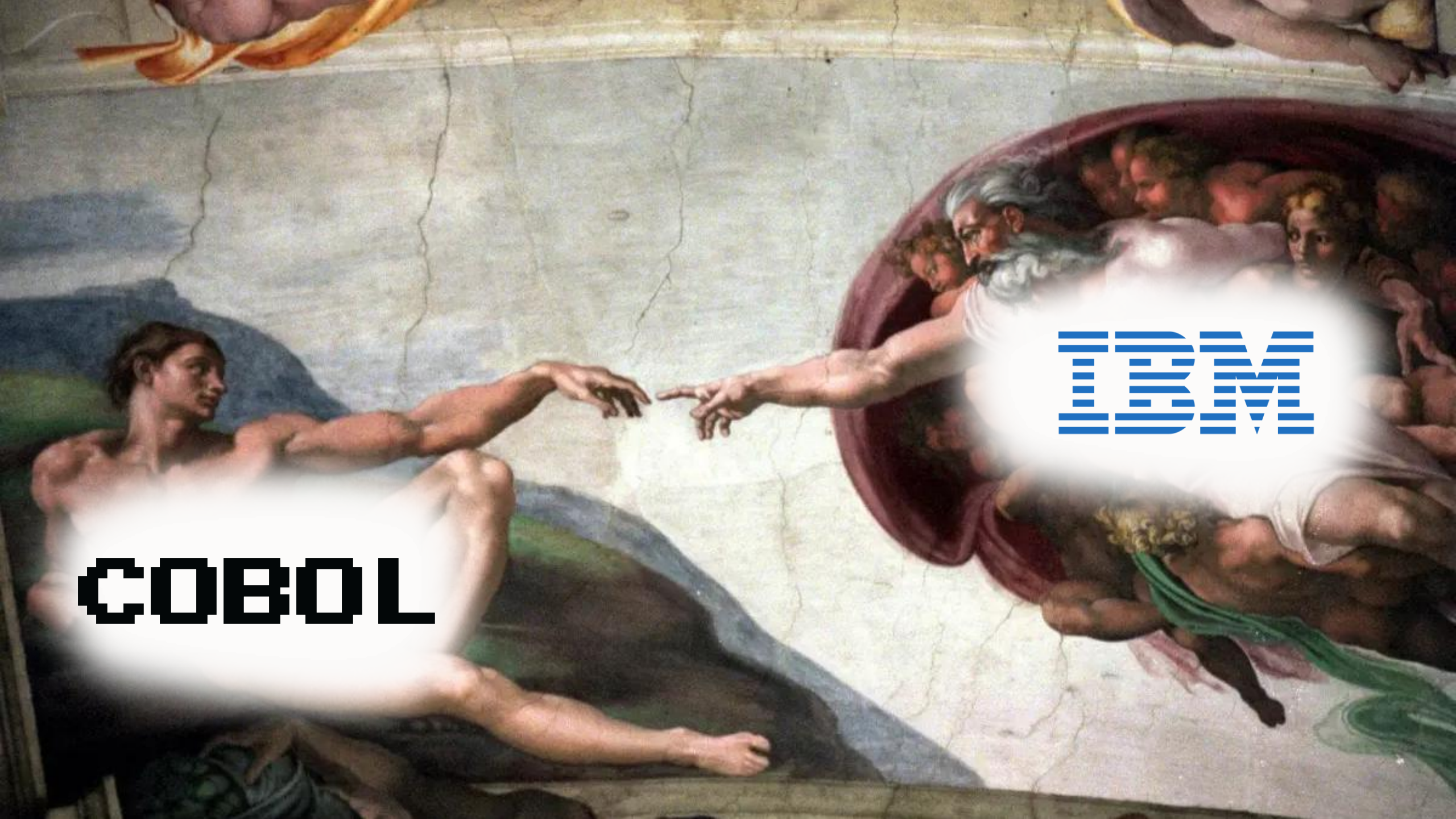



IBM
COBOL

things were simple




SLAYER



data management was simple



BANK.PROD.CUSTOMER.MASTER

D0000001001AMr	John	Smith	1982031512 King St	Perth	WA	6000	Australia	000500000000000125045
D0000001002AMrs	Sarah	Johnson	1975120188 Harbour Rd	Sydney	NSW	2000	Australia	000750000000000330010
U0000001003IDr	Michael	Nguyen	1968112245 Collins St	Melbourne	VIC	3000	Australia	001000000000000025075
D0000001004AMs	Emily	Brown	1990040507 Queen St	Brisbane	QLD	4000	Australia	000200000000000045000
D0000001005AMr	David	Wilson	1988071999 George St	Sydney	NSW	2000	Australia	000600000000001050025
U0000001006AProf	Linda	Taylor	1972033014 University Ave	Adelaide	SA	5000	Australia	001200000000007850000
D0000001007IMr	Robert	Anderson	1959091103 Market St	Hobart	TAS	7000	Australia	000150000000000000000
D0000001008AMrs	Jessica	Martin	19951225250 Ocean Dr	Perth	WA	6010	Australia	000450000000000128060
U0000001009AMr	Daniel	Thompson	1981110267 Victoria Rd	Darwin	NT	0800	Australia	000900000000002500000
D0000001010AMiss	Olivia	White	2001041718 Park Lane	Canberra	ACT	2600	Australia	000100000000000030000

"copy book"



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).

    05  CUSTOMER-STATUS           PIC X.
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).
        10 CUST-FIRST-NAME        PIC X(30).
        10 CUST-LAST-NAME         PIC X(30).

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).
           15 CUST-CITY           PIC X(30).
           15 CUST-STATE          PIC X(20).
           15 CUST-POSTCODE       PIC X(10).
           15 CUST-COUNTRY        PIC X(30).

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.
```



D0000001001AMr	John	Smith	19820315	12 King St	Perth	WA	6000	Australia	000500000 00000125045

```
01  CUSTOMER-RECORD.
    05  CUST-RECORD-TYPE          PIC X.                    D
        88 CUSTOMER-DETAIL        VALUE 'D'.
        88 CUSTOMER-UPDATE        VALUE 'U'.

    05  CUSTOMER-ID               PIC 9(10).                0000001001

    05  CUSTOMER-STATUS           PIC X.                    A
        88 CUST-ACTIVE            VALUE 'A'.
        88 CUST-INACTIVE          VALUE 'I'.

    05  CUSTOMER-NAME.
        10 CUST-TITLE             PIC X(10).                Mr
        10 CUST-FIRST-NAME        PIC X(30).                John
        10 CUST-LAST-NAME         PIC X(30).                Smith

    05  FILLER                    PIC X(20).

    05  CUSTOMER-DOB              PIC 9(8).                 19820315

    05  CUSTOMER-ADDRESSES.
        10 CUST-ADDRESS OCCURS 3 TIMES.
           15 CUST-ADDR-LINE1     PIC X(40).                12 King St
           15 CUST-CITY           PIC X(30).                Perth
           15 CUST-STATE          PIC X(20).                WA
           15 CUST-POSTCODE       PIC X(10).                6000
           15 CUST-COUNTRY        PIC X(30).                Australia

    05  CUSTOMER-CREDIT-LIMIT     PIC S9(7)V99 COMP-3.      000500000
    05  CUSTOMER-BALANCE          PIC S9(9)V99 COMP-3.      00000125045
```



```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT CUSTOMER-FILE
       ASSIGN TO 'BANK.PROD.CUSTOMER.MASTER'

FILE SECTION.
FD  CUSTOMER-FILE.
01  CUSTOMER-REC.
    05  CUST-RECORD-TYPE          PIC X.
         88 CUSTOMER-DETAIL        VALUE 'D'.
         88 CUSTOMER-UPDATE        VALUE 'U'.
    ...

MAIN-PARA.

   OPEN INPUT CUSTOMER-FILE

   PERFORM UNTIL END-OF-FILE
       READ CUSTOMER-FILE
            PERFORM PROCESS-RECORD
       END-READ
   END-PERFORM

   CLOSE CUSTOMER-FILE
   STOP RUN.
```



```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT CUSTOMER-FILE
       ASSIGN TO 'BANK.PROD.CUSTOMER.MASTER'

FILE SECTION.
FD  CUSTOMER-FILE.
01  CUSTOMER-REC.
    05  CUST-RECORD-TYPE          PIC X.
         88 CUSTOMER-DETAIL        VALUE 'D'.
         88 CUSTOMER-UPDATE        VALUE 'U'.
    ...

MAIN-PARA.

   OPEN INPUT CUSTOMER-FILE

   PERFORM UNTIL END-OF-FILE
       READ CUSTOMER-FILE
            PERFORM PROCESS-RECORD
       END-READ
   END-PERFORM

   CLOSE CUSTOMER-FILE
   STOP RUN.
```



```
INPUT-OUTPUT SECTION.
FILE-CONTROL.
   SELECT CUSTOMER-FILE
       ASSIGN TO 'BANK.PROD.CUSTOMER.MASTER'

FILE SECTION.
FD  CUSTOMER-FILE.
01  CUSTOMER-REC.
    05  CUST-RECORD-TYPE          PIC X.
         88 CUSTOMER-DETAIL        VALUE 'D'.
         88 CUSTOMER-UPDATE        VALUE 'U'.
    ...

MAIN-PARA.

   OPEN INPUT CUSTOMER-FILE

   PERFORM UNTIL END-OF-FILE
       READ CUSTOMER-FILE
            PERFORM PROCESS-RECORD
       END-READ
   END-PERFORM

   CLOSE CUSTOMER-FILE
   STOP RUN.
```



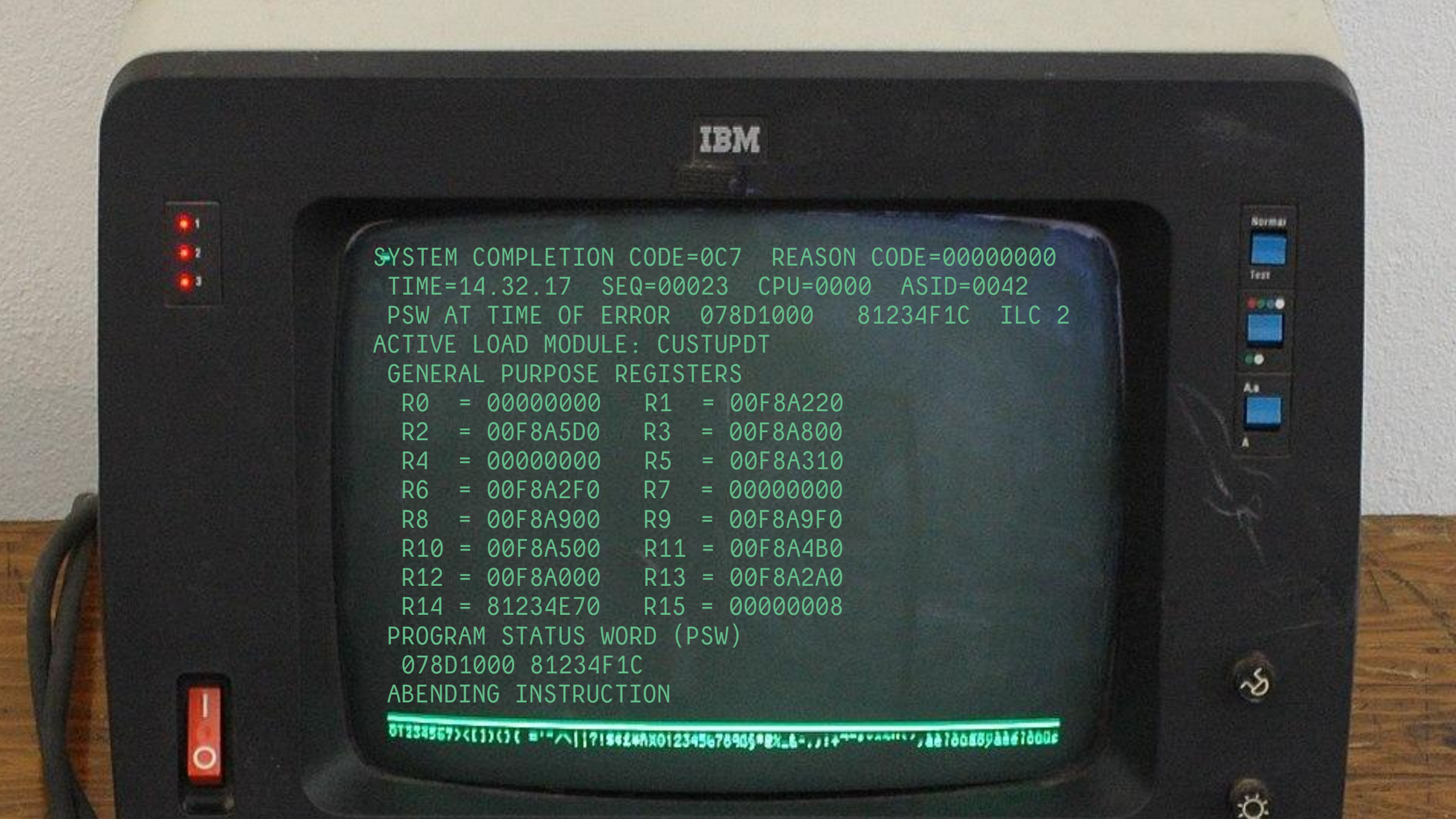
```
D0000001001AMr        John                Smith               19820315 12 King St                Perth          WA          6000        Australia          000500000 00000125045
```

19820315



19A20315




IBM
SYSTEM COMPLETION CODE=0C7 REASON CODE=00000000
TIME=14.32.17 SEQ=00023 CPU=0000 ASID=0042
PSW AT TIME OF ERROR 078D1000 81234F1C ILC 2
ACTIVE LOAD MODULE: CUSTUPDT
GENERAL PURPOSE REGISTERS
R0 = 00000000 R1 = 00F8A220
R2 = 00F8A5D0 R3 = 00F8A800
R4 = 00000000 R5 = 00F8A310
R6 = 00F8A2F0 R7 = 00000000
R8 = 00F8A900 R9 = 00F8A9F0
R10 = 00F8A500 R11 = 00F8A4B0
R12 = 00F8A000 R13 = 00F8A2A0
R14 = 81234E70 R15 = 00000008
PROGRAM STATUS WORD (PSW)
078D1000 81234F1C
ABENDING INSTRUCTION

```
SYSTEM COMPLETION CODE=0C7  REASON CODE=00000000
 TIME=14.32.17  SEQ=00023  CPU=0000  ASID=0042
 PSW AT TIME OF ERROR  078D1000   81234F1C  ILC 2
ACTIVE LOAD MODULE: CUSTUPDT
 GENERAL PURPOSE REGISTERS
  R0  = 00000000   R1  = 00F8A220
  R2  = 00F8A5D0   R3  = 00F8A800
  R4  = 00000000   R5  = 00F8A310
  R6  = 00F8A2F0   R7  = 00000000
  R8  = 00F8A900   R9  = 00F8A9F0
  R10 = 00F8A500   R11 = 00F8A4B0
  R12 = 00F8A000   R13 = 00F8A2A0
  R14 = 81234E70   R15 = 00000008
 PROGRAM STATUS WORD (PSW)
  078D1000 81234F1C
 ABENDING INSTRUCTION
```

```
SYSTEM COMPLETION CODE=0C7  REASON CODE=00000000
 TIME=14.32.17  SEQ=00023  CPU=0000  ASID=0042
 PSW AT TIME OF ERROR  078D1000   81234F1C  ILC 2
ACTIVE LOAD MODULE: CUSTUPDT
 GENERAL PURPOSE REGISTERS
  R0  = 00000000   R1  = 00F8A220
  R2  = 00F8A5D0   R3  = 00F8A800
  R4  = 00000000   R5  = 00F8A310
  R6  = 00F8A2F0   R7  = 00000000
  R8  = 00F8A900   R9  = 00F8A9F0
  R10 = 00F8A500   R11 = 00F8A4B0
  R12 = 00F8A000   R13 = 00F8A2A0
  R14 = 81234E70   R15 = 00000008
 PROGRAM STATUS WORD (PSW)
  078D1000 81234F1C
 ABENDING INSTRUCTION
```

there were no* corrupt databases

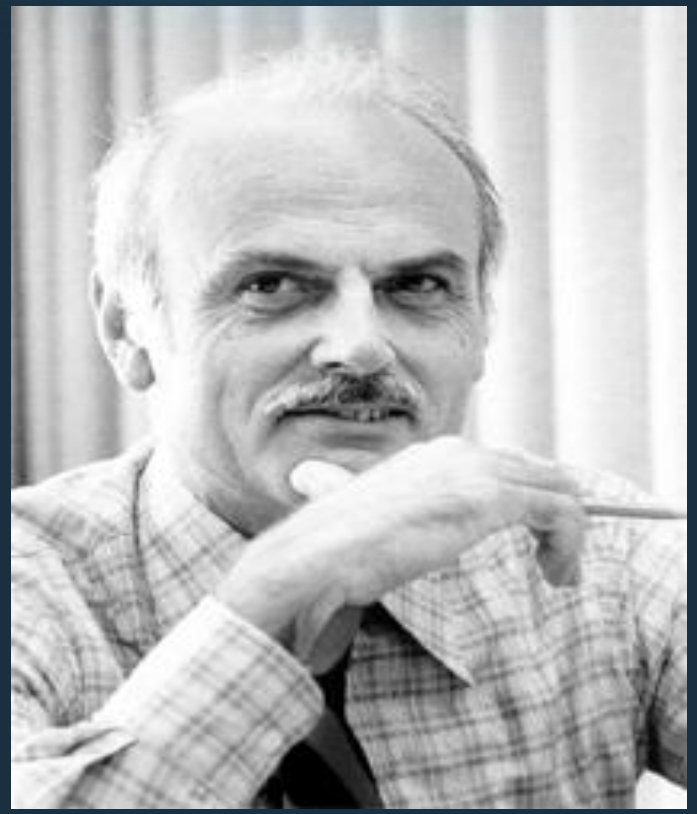


a corrupt database …



.. was a down database :-)





relational databases



data types

mandatory

declarative data integrity

uniqueness

relationships



for us it was awesome !



our APIs became simple



```java
public static void newEmp(Connection conn,
                          int empno,
                          String ename,
                          String job,
                          Integer mgr,
                          Date hiredate,
                          double sal,
                          int deptno) throws SQLException {

    String sql =
        "INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, deptno) " +
        "VALUES (?, ?, ?, ?, ?, ?, ?)";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, empno);
        pstmt.setString(2, ename);
        pstmt.setString(3, job);
        if (mgr != null)
            pstmt.setInt(4, mgr);
        else
            pstmt.setNull(4, java.sql.Types.INTEGER);
        pstmt.setDate(5, hiredate);
        pstmt.setDouble(6, sal);
        pstmt.setInt(8, deptno);
        pstmt.executeUpdate();
    }
}
```


RDBMS
ISAM
VSAM
IMS





Oracle

1979





McDonald's conjecture

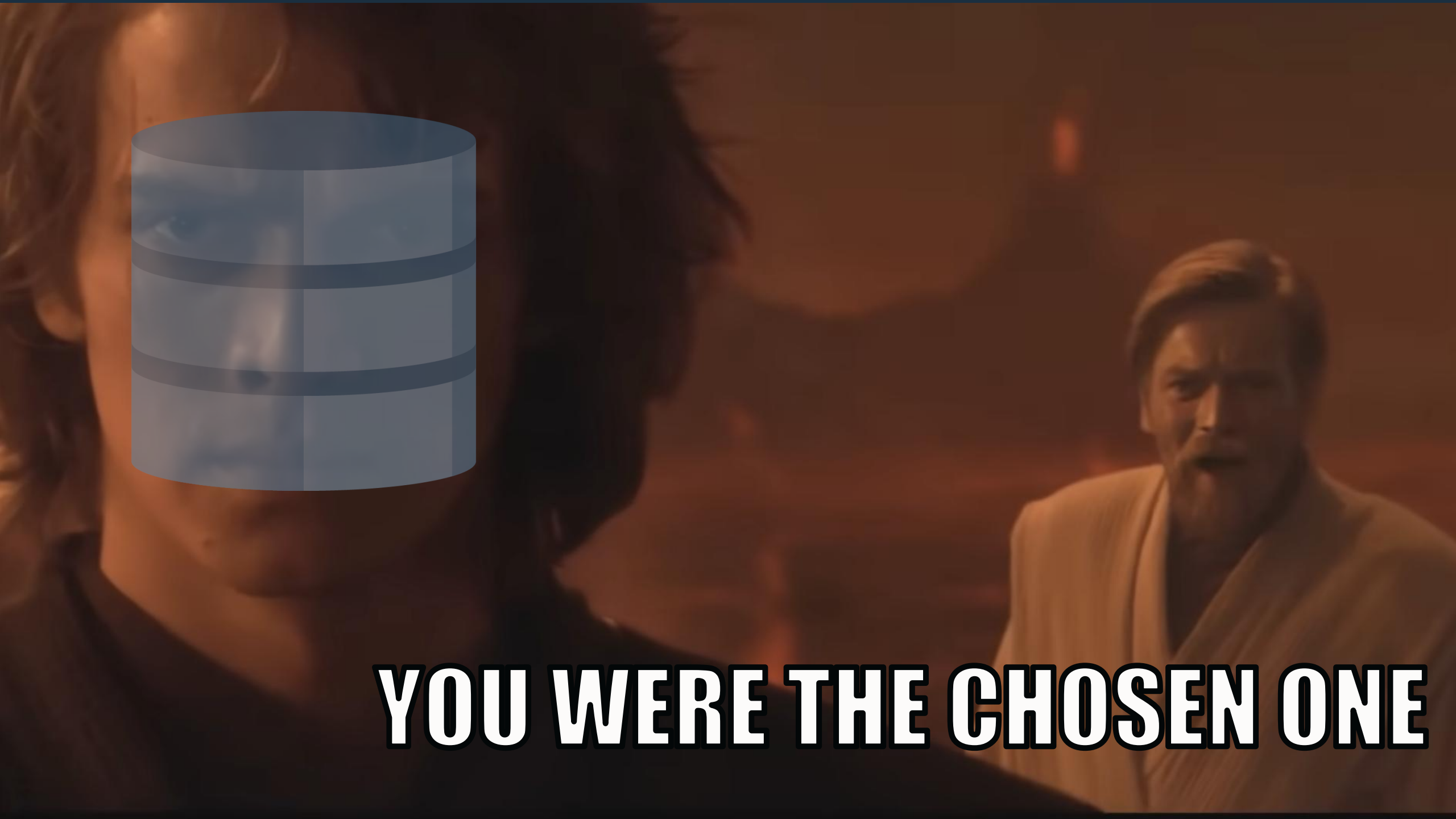


EVERY DATABASE



ie, contain corrupt data




YOU WERE THE CHOSEN ONE

data types

mandatory

1990

declarative data integrity

uniqueness

relationships



but that was it …



data integrity has stalled



the real world is more than this...



```
SQL> select * from emp order by deptno, empno;
```

EMPNO	ENAME	JOB	MGR	HIREDATE	SAL	COMM	DEPTNO
7782	CLARK	MANAGER	7839	09-JUN-81	2450		10
7839	KING	DEPT HEAD		17-NOV-81	5000		10
7934	MILLER	CLERK	7782	23-JAN-82	1300		10
7369	SMITH	CLERK	7902	17-DEC-80	800		20
7566	JONES	MANAGER	7839	02-APR-81	2975		20
7788	SCOTT	ANALYST	7566	09-DEC-82	3000		20
7876	ADAMS	CLERK	7788	12-JAN-83	1100		20
7902	FORD	DEPT HEAD	7566	03-DEC-81	3000		20
7499	ALLEN	SALESMAN	7698	20-FEB-81	1600	300	30
7521	WARD	SALESMAN	7698	22-FEB-81	1250	500	30
7654	MARTIN	SALESMAN	7698	28-SEP-81	1250	1400	30
7698	BLAKE	DEPT HEAD	7839	01-MAY-81	2850		30
7844	TURNER	SALESMAN	7698	08-SEP-81	1500		30
7900	JAMES	CLERK	7698	03-DEC-81	950		30



```
SQL> select * from emp order by deptno, empno;

     EMPNO ENAME      JOB              MGR HIREDATE         SAL       COMM     DEPTNO
---------- ---------- --------- ---------- --------- ---------- ---------- ----------
      7782 CLARK      MANAGER         7839 09-JUN-81       2450                    10
      7839 KING       DEPT HEAD            17-NOV-81       5000                    10
      7934 MILLER     CLERK           7782 23-JAN-82       1300                    10

      7369 SMITH      CLERK           7902 17-DEC-80        800                    20
      7566 JONES      MANAGER         7839 02-APR-81       2975                    20
      7788 SCOTT      ANALYST         7566 09-DEC-82       3000                    20
      7876 ADAMS      CLERK           7788 12-JAN-83       1100                    20
      7902 FORD       DEPT HEAD       7566 03-DEC-81       3000                    20

      7499 ALLEN      SALESMAN        7698 20-FEB-81       1600        300         30
      7521 WARD       SALESMAN        7698 22-FEB-81       1250        500         30
      7654 MARTIN     SALESMAN        7698 28-SEP-81       1250       1400         30
      7698 BLAKE      DEPT HEAD       7839 01-MAY-81       2850                    30
      7844 TURNER     SALESMAN        7698 08-SEP-81       1500                    30
      7900 JAMES      CLERK           7698 03-DEC-81        950                    30
```



"No employee can have a salary higher than the salary of their head of department"



cannot be done declaratively



"No problem!"

just a simple change to our API



```
//-----------------------------------------------------
// Check department head salary rule
//-----------------------------------------------------
private static void validateDeptHeadSalary(Conn
                                           int
                                           doub


    String sql =
        "SELECT sal FROM emp " +
        "WHERE deptno = ? AND job = 'DEPT HEAD'

    try (PreparedStatement pstmt = conn.prepar
        pstmt.setInt(1, deptno);
        ResultSet rs = pstmt.executeQuery();
        if (rs.next()) {
            double headSalary = rs.getDouble(1)
            if (salary > headSalary) {
                throw new SQLException(
                    "Salary exceeds DEPT HEAD s
                );
            }
        }
    }
}
```

```
// ---------------------------------------------
// Insert new employee
// ---------------------------------------------
public static void newEmp(Connection conn,
                          int empno,
                          String ename,
                          String job,
                          Integer mgr,
                          Date hiredate,
                          double sal,
                          int deptno) throws SQLException {


    validateDeptHeadSalary(conn, deptno, sal);


    String sql =
        "INSERT INTO emp (empno, ename, job, mgr, hiredate, sal, deptno) " +
        "VALUES (?, ?, ?, ?, ?, ?, ?)";

    try (PreparedStatement pstmt = conn.prepareStatement(sql)) {
        pstmt.setInt(1, empno);
        pstmt.setString(2, ename);
        pstmt.setString(3, job);
        if (mgr != null)
            pstmt.setInt(4, mgr);
        else
            pstmt.setNull(4, java.sql.Types.INTEGER);
        pstmt.setDate(5, hiredate);
        pstmt.setDouble(6, sal);
        pstmt.setInt(8, deptno);
        pstmt.executeUpdate();
    }
}
```

let's explore that



"Engineers call them edge cases. I call them: What our users do."

- Noah Sussman



it gets very complicated ...



... very very fast

vd3


I'VE IMPLEMENTED THE DATA QUALITY CHECKS IN THE APP
SO WE WON'T HAVE ANY MORE DATA CORRUPTION RIGHT?
RIGHT?

we're not even close :-(



Hofstadter's law:

It always takes longer than you expect,
even when you take into account Hofstadter's law



common uses cases .... ✓

edge cases ... ✓

robust APIs ... ✓

unit tests pass ... ✓

"all of your databases are corrupt"



benefit of the doubt



no adhoc access to the data

"we only allow the APIs"



no "maintenance" API

aka, the hacker's delight



strict process controls (eg ITIL)



one tiny winy remaining issue ...



NONE OF THIS WORKS !

the most common gotcha





it seems so easy



```
select  'It is bad when...'
from   emp e
where job = 'DEPT HEAD'
and exists (
   select '... there is too high a salary'
   from    emp
   where   deptno = e.deptno
   and     sal > e.sal )
```



that's all we want !



we knew this struggle



SQL 1992



rules that span rows or tables



www.contrib.andrew.cmu.edu/~shadow/sql/sql1992.txt

4.10.4 Assertions

An assertion is a named constraint that may relate to the content of individual rows of a table, to the entire contents of a table, or to a state required to exist among a number of tables.

An assertion is described by an assertion descriptor. In addition to the components of every constraint descriptor an assertion descriptor includes:

- the <search condition>.

An assertion is satisfied if and only if the specified <search condition> is not false.

```
create ASSERTION check
   ( <Boolean expression> )
```



```
create ASSERTION check
   not exists (
      select  'It is bad when...'
      from  emp e
      where job = 'DEPT HEAD'
      and exists (
        select '... there is too high a salary'
        from   emp
        where  deptno = e.deptno
        and    sal > e.sal )
   )
```



examples



easy to get (over) excited :-)



alter table add unique ...

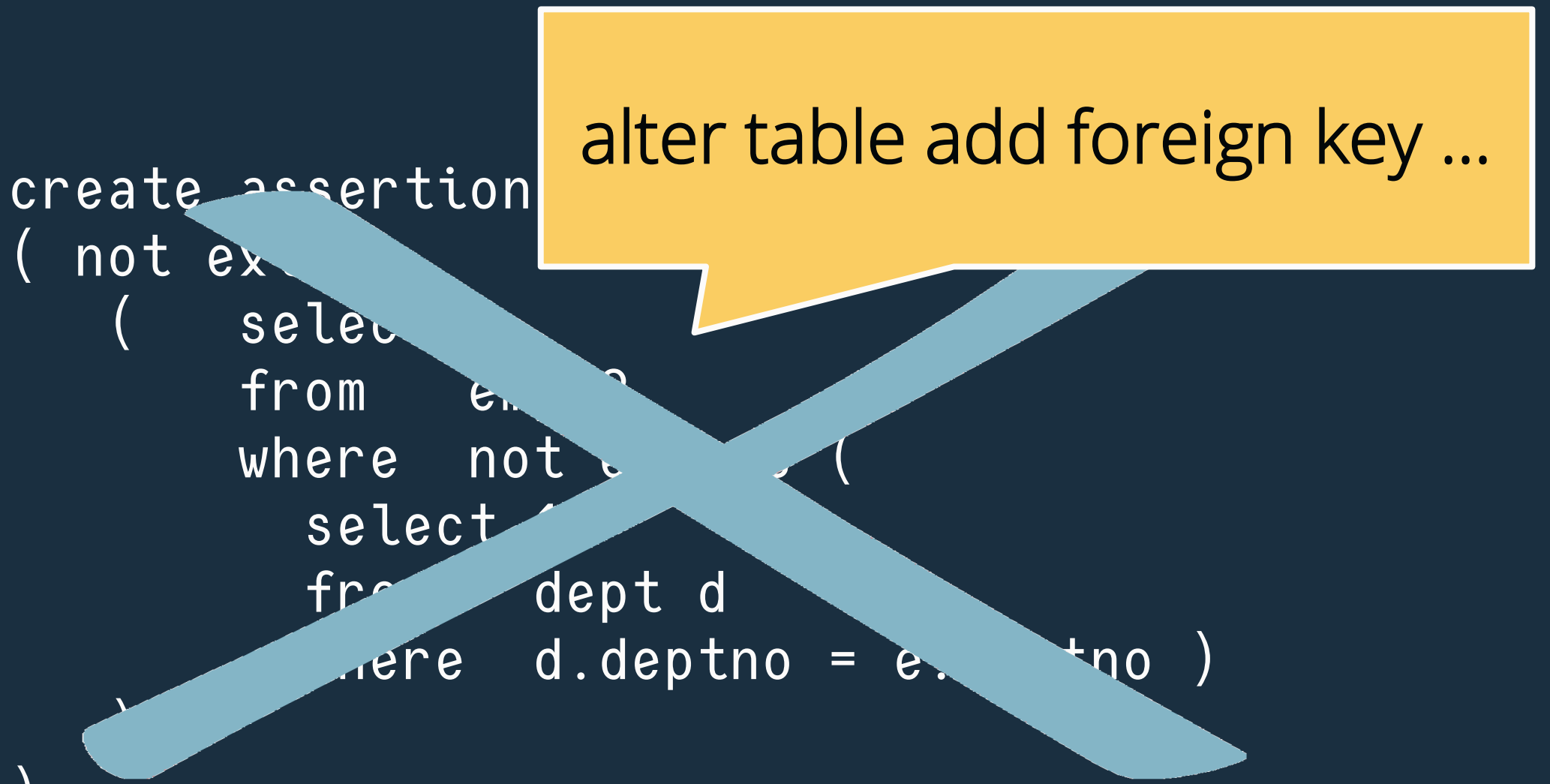
```
create a      tion unique_
( not exists
    (    select 1
         from   emp e      p e2
         where  e1.      name      name
         and    e1.empno !=      empno
    )
);
```



alter table add check ...

```
create asse      on valid  ate  ge check
( not exists
    (    select 1
         from   em
         where  tart_date    nd_date
    )
);
```




alter table add foreign key ...
create assertion
(not ex
(sele
from e
where not
select
fr dept d
here d.deptno = e. tno)
);

what about performance ?



issue 1

"Every DML means all of those SQL's in all of those assertions?"



No …



… track potential impact



```
SQL> create assertion MUST_HAVE_PRESIDENT check
  2  ( exists
  3    ( select 1
  4      from   emp
  5      where  job = 'PRESIDENT'
  6    )
  7  );
```

delete a row ...

insert a row ...

update a row ...

to/from PRESIDENT



issue 2

"Aren't we back to checking the entire table again?"

No … we can cheat :-)

"incremental validation"



```
SQL> create assertion DEPT_MUST_HAVE_ASSISTANT check
  2  ( not exists
  3    ( select 1
  4      from   emp e1
  5      where  not exists (
  6        select 1 from emp e2
  7        where  e2.deptno = e1.empno
  8        and    e2.job = 'ASSISTANT' )
  9  ));
```

```
"validate" assertion DEPT_MUST_HAVE_ASSISTANT check
 ( not exists
    ( select 1
      from   emp e1
       ( select *
         from   <affected rows>
         where  <before or after> job = 'ASSISTANT' )
      where  not exists (
        select 1 from emp e2
        where  e2.deptno = e1.empno
        and    e2.job = 'ASSISTANT' )
 ));
```

```
SQL> select
  2    assertion_name,
  2    validation_type,
  3    validation_event
  4  from user_assertion_dependencies;

ASSERTION_NAME                 VALIDATION_TYPE VALIDATION_EVENT
------------------------------ --------------- ----------------------
MUST_HAVE_PRESIDENT            COMPLETE        ROWS DELETED OR UPDATED
DEPT_MUST_HAVE_ASSISTANT       FAST            ROWS INSERTED OR UPDATED
```



better than you can do

vd9



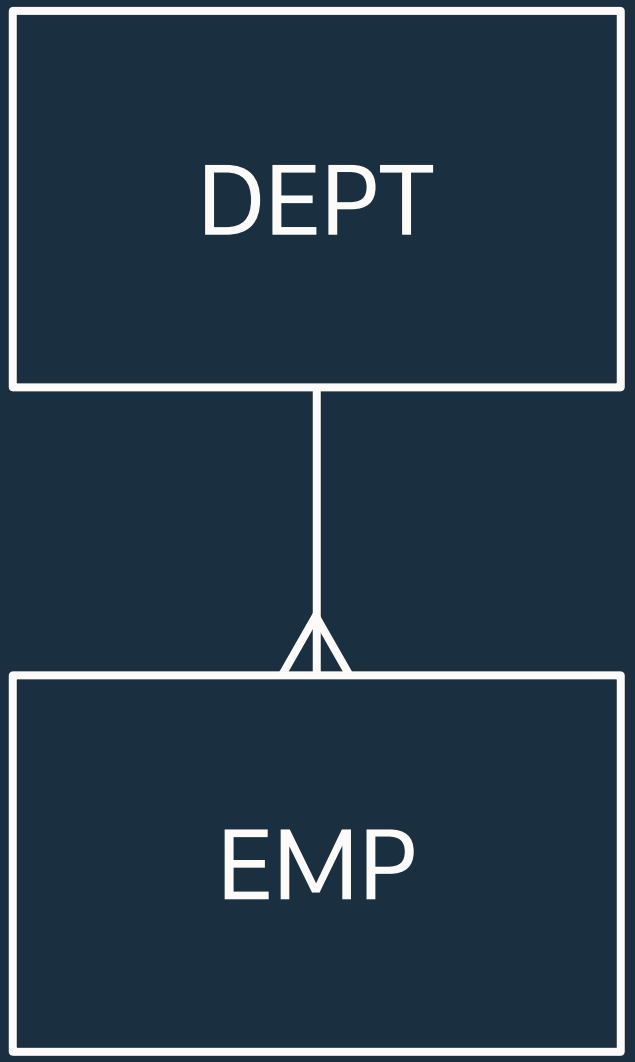
what about concurrency ?





session 1

```
SQL> delete from DEPT
  2  where dname = 'HR';
```



session 2

```
SQL> insert into EMP
  2  values (..., 'HR')
```

(blocked)



same with assertions



```
SQL> create assertion MUST_HAVE_PRESIDENT check
  2  ( exists
  3    ( select 1
  4      from   emp
  5      where  job = 'PRESIDENT'
  6    )
  7  );
```



session 1

```
SQL> delete from EMP
  2  where job = 'PRESIDENT'
```

(commit or rollback)

EMP

session 2

```
SQL> delete from EMP
  2  where job = 'PRESIDENT'
```

(blocked)

table rows not locked



some negatives about … negatives







"Every manager must have an assistant"

"There cannot be a ...

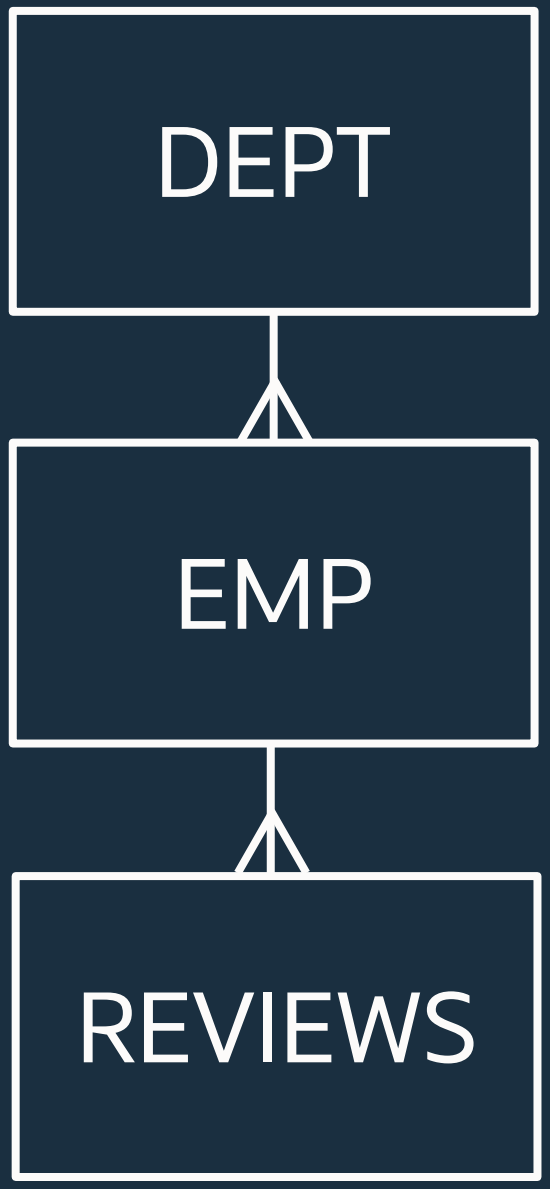
manager

have an assistant"

"who does not..."

```
create assertion NO_INVALID_MANAGERS
check (
    not exists (
        select 'bad manager'
        from  emp e1
        where job = 'MANAGER'
        and not exists (
                select 'cannot find an assistant'
                  from emp e2
                 where e2.deptno = e1.deptno
                   and job = 'ASSISTANT')
    )
);
```




DEPT
EMP
REVIEWS

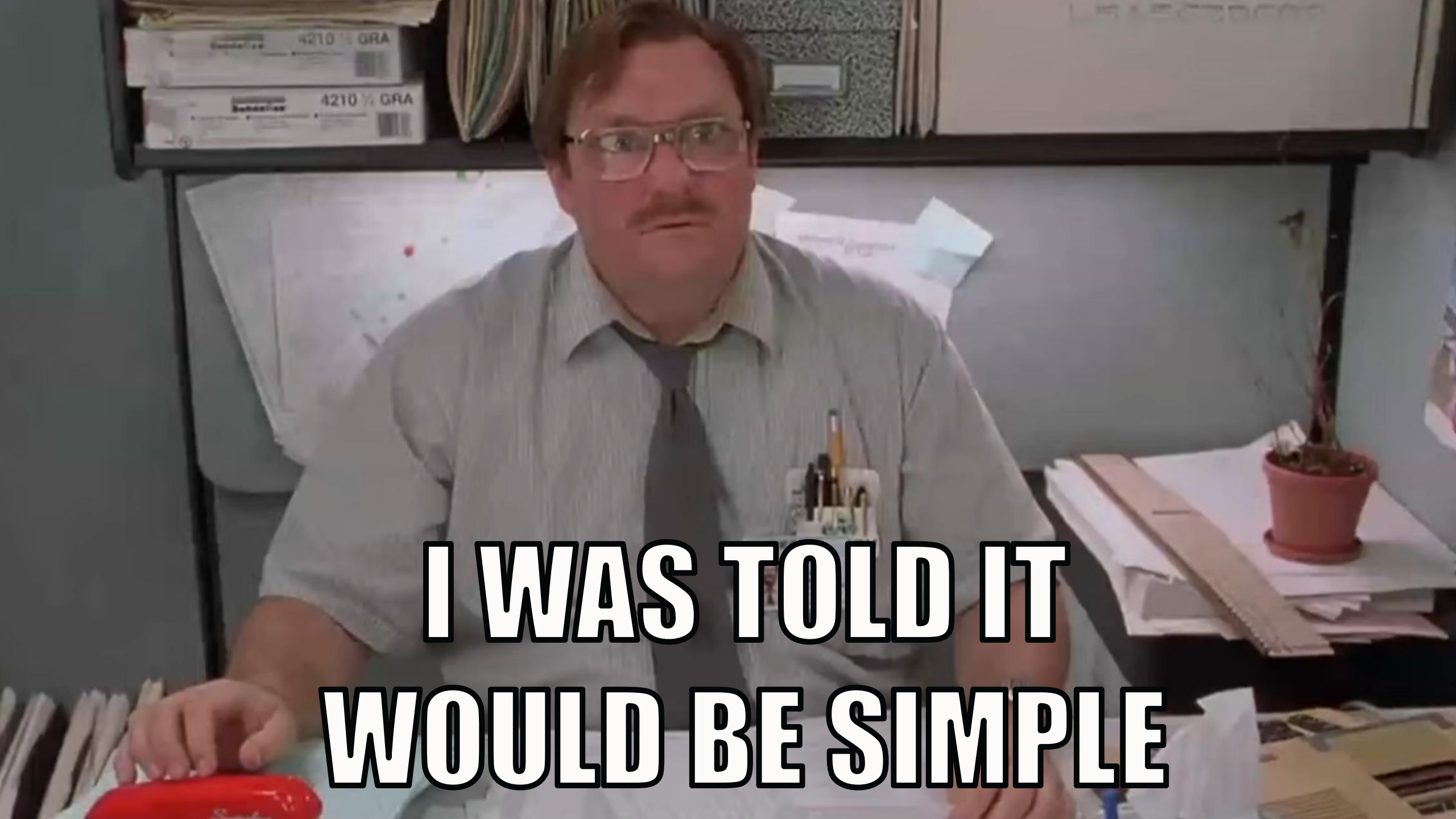
"Every department must have an employee with a clean performance review"



"There can not be a department that does not have an employee who has not been given a warning"

```
create assertion NO_INVALID_DEPARTMENTS
check (
   not exists (
      select 'bad dept'
      from  dept d
      where not exists (
          select 'cannot find a bad person'
          from emp e
          where e.deptno = e.deptno
          and not exists (
             select 'warning'
             from    reviews
             where   empno = e.empno
             and     results = 'WARNING'
           )
       )
   )
);
```




I WAS TOLD IT
WOULD BE SIMPLE

ALL-SATISFY workaround



"Every manager must have an assistant"



```
create assertion NO_INVALID_MANAGERS
check (
    ALL (
        select deptno
        from  emp
        where job = 'MANAGER' ) m
    SATISFY (
        exists (
            select 'found an assistant'
            from  emp
            where deptno = m.deptno
            and   job = 'ASSISTANT')
            )
    )
);
```

Note: syntax "sugar"



- <= 3 levels
- equi-joins
- tables only
- deterministic

initial implementation

- no outer join
- no aggregation



wrap up



two key points



1) data integrity is hard

much much harder than you think



corollary: your database is corrupt

even if you think it's not :-)



2) assertions are a game changer



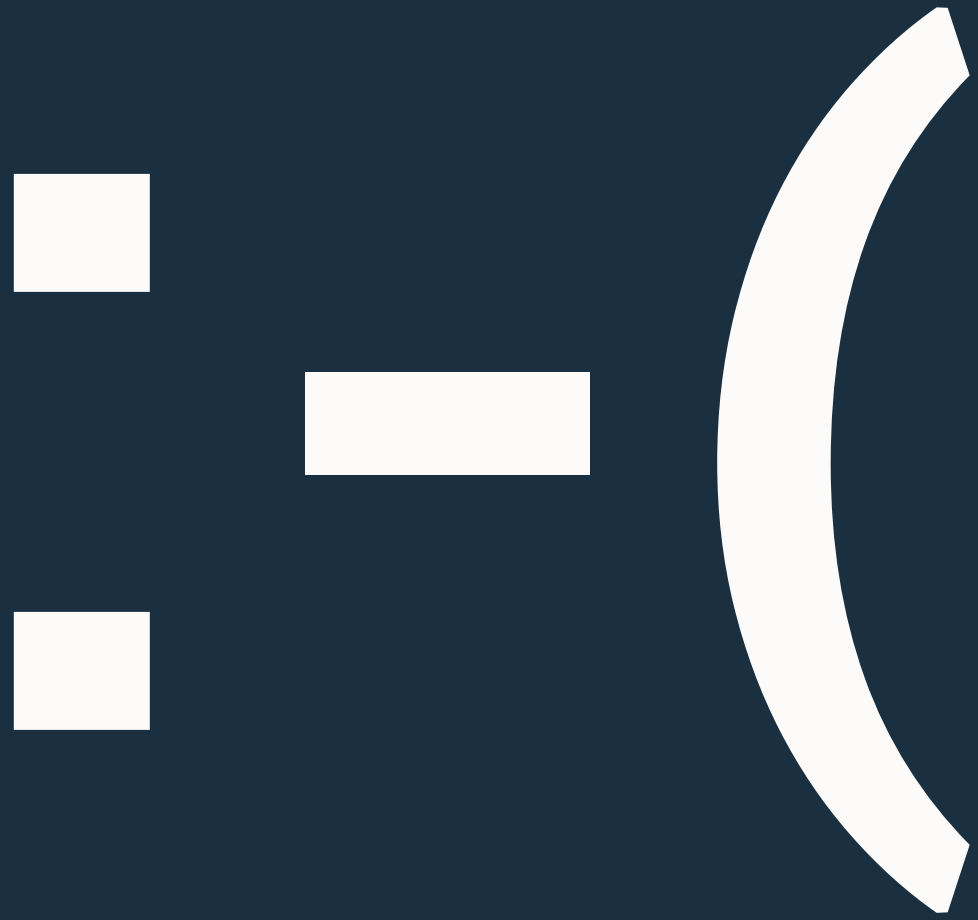
we're not fully there yet ...

… but we've started the journey



the future looks awesome






Thank you!