# Föderierung von Dateien: Finden, Verwalten und Nutzen von Metainformationen [*]

Michael Höding

Otto-von-Guericke-Universität Magdeburg
Institut für Technische Informationssysteme
Postfach 4120, D–39016 Magdeburg
E-mail: `hoeding@iti.cs.uni-magdeburg.de`
Tel.: `++49-391-67-12220` Fax: `++49-391-67-12020`

Im Rahmen des Projektes **SIGMA**$_{FDB}$ sind Fragestellungen der Unterstützung einer unternehmensweiten Informations- und Kommunikationsinfrastruktur zur durchgängigen Unternehmensorganisation zu untersuchen. Insbesondere geht es um Konzepte zur informationstechnischen Unterstützung des Fabrikplanungsprozesses. Wir gehen davon aus, daß der Einsatz föderierter Systeme ein geeigneter Ansatz ist, um vorhandene bzw. neu hinzukommende Applikationen korrekt zu integrieren und die Erhaltung eines konsistenten globalen Datenbestandes zu gewährleisten. Bei einer ersten Anforderungsanalyse und bei der Analyse der vorhandenen, teilweise gekoppelten Spezialapplikationen hat sich insbesondere die Forderung nach der Integration von auf spezifischen Dateiformaten basierenden Daten herauskristallisiert. Zu diesem Thema werden im vorliegenden Papier wichtige Fragestellungen aufgeworfen und erste Lösungsansätze skizziert. Nach einer weitergehenden Motivation des Themas wird die zentrale Frage des Findens und Aufbereitens der Metainformation für verschiedene Klassen von Dateien diskutiert. Im weiteren werden Überlegungen zur möglichen Realisierung eines solchen heterogenen Informationssystems vorgestellt. Abschließend werden die Ergebnisse kurz zusammengefaßt und ein Ausblick auf zukünftige Arbeiten gegeben.

## 1 Einführung und Motivation

Komplexe Informationssysteme in Wirtschaft und Verwaltung bestehen zumeist aus einer Reihe spezieller Applikationsprogramme, die ihre Daten mittels verschiedenartiger Datenhaltungssysteme speichern. Dies resultiert u. a. aus der Tatsache, daß vorhandene homogene monolithische Systeme nicht flexibel genug an spezielle Nutzeranforderungen angepaßt werden konnten und deshalb durch Beistellung spezieller Lösungen erweitert wurden. Andere heterogene Systeme bestehen vollständig aus einer Menge spezieller Applikationen, die unabhängig voneinander entwickelt wurden. Eine moderne Ausprägung dieses Ansatzes ist das Client-Server-Konzept, bei dem alle Client-Applikationen über eine einheitliche Schnittstelle auf die gemeinsamen, von einem Datenbankserver verwalteten Informationen zugreifen. Im Gegensatz dazu basiert die Datenhaltung bei einer Reihe von heterogenen Informationssystemen auf applikationsspezifischen Dateiformaten und Konvertern zwischen den Dateien, die applikationsübergreifende Daten zum Teil redundant speichern. Solche Lösungen haben sich in der Vergangenheit vor allem im PC-Bereich etabliert. Ein gutes Beispiel dafür ist das Fabrikplanungssystem LOGAP[1], das Teile des Fabrikplanungsprozesses unterstützt [HTJ+95, FS94]. Einige Nachteile derartiger Lösungen werden aus verschiedenen Gründen nicht sofort offensichtlich. Der LOGAP ist eine typische Einplatzlösung, bei der die Synchronisation mehrerer Benutzer, d.h. mehrerer gleichzeitig arbeitender Applikationsprogramme, durch den Bediener durchgeführt wird. Dieser Punkt erweist sich in der heutigen Zeit der zunehmenden Verbreitung

[*] Diese Arbeit wurde gefördert vom Land Sachsen-Anhalt unter FKZ: 1987A/0025 "Föderierung heterogener Datenbanksysteme und lokaler Datenhaltungskomponenten zur systemübergreifenden Integritätssicherung" – "**SIGMA**$_{FDB}$ (**S***chema* **I***ntegration and* **G***lobal integrity* **M***aintenance* **A***pproach for* **F***ederated* **D***ata* **B***ases*)"

[1] **Log**istik**a**rbeits**p**latz. Der LOGAP ist die prototypische Realisierung eines Fabrikplanungssystems, das insbesondere Layoutplanungs- und Materialflußoptimierungsaufgaben unterstützt. Die dabei eingesetzten Spezialapplikationen wurden um Konvertierungsprogramme ergänzt und erlauben somit teilweise iteratives Arbeiten. Der LOGAP stellt derzeit eine Laborlösung dar, bei der Probleme beim Zusammenspiel der Komponentenapplikationen durch das Know-how des Bedieners umgangen werden.

von Netzwerklösungen und Netzwerkdateisystemen als problematisch, läßt doch die scheinbar einfache Portierung der etablierten Systeme und deren gleichzeitige Nutzung durch mehrere Anwender unerwartete Defizite offensichtlich werden. Es muß jedoch beachtet werden, daß derartige Informationssysteme oft wichtige Teile des Know-how der Unternehmen enthalten und nicht ohne weiteres ersetzt werden können. Deshalb gilt es, Wege zu finden, um diese Systeme und ihre Daten auch in Zukunft nutzen zu können, sie den neuen Erfordernissen und Möglichkeiten anzupassen und sie einfach um zusätzliche Komponenten erweiterbar zu machen.

Als vielversprechender Ansatz für die Integration heterogener Systeme wird in den letzen Jahren zunehmend der Einsatz föderierter Datenbanksysteme diskutiert [HM85]. Dabei werden Anwendungsdaten, die durch verschiedene heterogene Datenbanksysteme verwaltet werden, durch den Aufsatz eines föderierten Datenbankmanagementsystems (FDBMS) zusammengefaßt (vgl. Bild 1). Entsprechend der

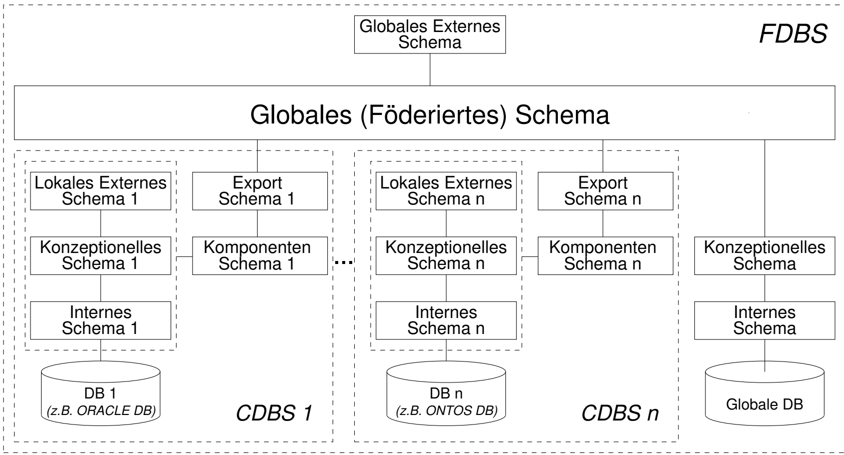


Abbildung 1: 5-Ebenen Referenzarchitektur nach [SL90]

5-Ebenen-Referenzarchitektur nach [SL90] werden die lokalen Schemata der Komponentendatenbanken in das föderierte Schema des FDBMS umgesetzt. Das FDBMS übernimmt die Kontrolle vorhandener Redundanzen, die Aufdeckung von Inkonsistenzen und ermöglicht über eine externe Schnittstelle den Zugriff auf alle Daten des heterogenen Informationssystems. Auf die Grundlagen föderierter Datenbanksysteme soll an dieser Stelle nicht weiter eingegangen werden. Sie wurden in großer Ausführlickeit u.a. in [SL90, LMR90, HTJ+95] beschrieben. In diesem Papier möchten wir das Augenmerk insbesondere auf den Schritt der Datenmodelltransformation richten, der der Überwindung der Datenmodellheterogenität dient. Ein ähnlicher Schritt ist auch bei der Integration anwendungsspezifischer Dateien vorzunehmen, kann man doch das spezifische Dateiformat als lokales Datenmodell betrachten. Deshalb nutzen wir im weiteren den Begriff Datenmodelltransformation für diesen Entwurfsschritt.

Um diesen, wegen der großen Anwendungsabhängigkeit sehr aufwendigen Entwurfsschritt zu vereinfachen, gilt es Wege zu finden, den Entwurfsprozeß für typische Dateiformate zu unterstützen. Als zentrale Frage ist dabei das Finden und Aufbereiten der Metainformationen für die anwendungsspezifischen Dateien zu nennen. Als Ergebnis dieses Entwurfsschrittes liegt ein zu den lokalen Dateien äquivalentes Schema auf Basis eines etablierten Datenmodells vor. Die folgenden Entwurfsschritte der Schemaintegration und Ableitung externer Schemata können entsprechend der für verschiedene Datenmodelle vorgeschlagenen Methodiken durchgeführt werden (vgl. u.a. [BLN86, RPRG94, Sch95]).

# 2 Finden und Ableiten von Metainformationen

Für die Integration von Dateien ist der Schritt der Datenmodelltransformation von besonderer Bedeutung. Dabei geht es darum, für die speziellen Dateiformate der Applikationen äquivalente lokale Schemata abzuleiten. Wenn möglich sollte dazu das durch das FDBMS genutzte kanonische Datenmodell als Zieldatenmodell dienen, womit eine weitere Datenmodelltransformation vermieden werden kann.

Anders als bei der Datenmodelltransformation zwischen einigen bekannten Datenmodellen, die bereits seit Jahren Gegenstand der Datenbankforschung ist, kann sich die Integration von Dateien als sehr aufwendiges und teilweise unmögliches Unterfangen herausstellen.

Bei ersten Untersuchungen verschiedener Dateiformate, u.a. aus dem Bereich der oben erwähnten Anwendung aus der Fabrikplanung, hat sich eine Unterscheidung folgender Klassen von Dateiformaten ergeben. Für die Dateiformate jeder Klasse ist es möglich, ein ähnliches Vorgehen für die Integration im Rahmen einer Datenbankföderation abzuleiten. Die zentrale Frage ist dabei das Finden der Metainformationen, auf die im folgenden näher eingegangen werden soll. Ziel ist die vollständige, korrekte und möglicherweise automatische Ableitung der Metainformationen zu den lokal genutzten Dateien. Metadaten, also Informationen über die Art, Struktur und Abspeicherung der Daten in Dateien, können bei Applikationen mit eigenen Datenhaltungssystemen sowohl in die Applikation "hineinprogrammiert" als auch in den Dateien enthalten sein. Wir finden weder eine konsequente Trennung von Daten und Metadaten noch eine Trennung von interner und konzeptioneller Ebene wie bei modernen Datenbanksystemen vor.

- **Relationale Tabellen ähnlich xBase-Dateien**
  Diese Dateiformate werden im PC-Bereich häufig genutzt. Sie sind eng an das relationale Datenmodell angelehnt. Aus diesem Grund, und weil sich um einen "Quasi-Standard" handelt, gestaltet sich die Integration derartiger Dateien vergleichsweise einfach. Metainformation ist sowohl in den Dateinamen als auch in den Tabellendateien selbst enthalten. Der Dateiname entspricht dem Relationennamen. Anhand der Extension des Dateinamens kann man unterscheiden, ob eine Tabellendatei Nutzerdaten oder andere Information, wie etwa Zugriffspfade enthält. Eine typische Datentabelle enthält in einem Anfangsblock Informationen (Metainformationen) über die Struktur der Datensätze. Anhand dieser Informationen ist es einfach möglich, ein äquivalentes Relationenschema mit entsprechenden Mappinginformationen abzuleiten. Auch die sinnvolle Nutzung der Zugriffspfade wird möglich.

- **Stark strukturierte Texte**
  Typische Vertreter dieser Klasse sind Dateien im bekannten DXF-Format[2], in dem geometrische Objekte durch fest vorgegebene Strukturen beschrieben werden. Dabei werden jeder Menge von Attributwerten die zugehörigen Metainformationen als Schlüsselworte vorangestellt. Die vollständige Struktur kann nur durch Kenntnis der Semantik der Schlüsselworte rekonstruiert werden. Basiert die Datei auf einem Standard-Dateiformat zu dem eine Grammatik existiert, ist eine Transformation durch Parser möglich. Anderenfalls ist eine entsprechende Grammatik zu finden.

- **Schwach strukturierte Texte**
  Als schwach strukturierte Texte bezeichnen wir Dateien, die neben einer Reihe informaler Informationen auch Daten in formaler Form enthalten. Beispiele dafür sind Telefonnummern in WWW-Seiten oder Preise und Maschinendaten aus Produktkatalogen. Solche Informationen werden zumeist (aber nicht sicher) mit einem Schlüsselwort aus einer vorgegebenen Menge (z.B. tel., telefon, phone, tel.-nr, etc.) eingeleitet oder repräsentieren sich durch ihr Format. Eine Integration solcher Dateien in Föderationen ist, wie sich am Beispiel der Produktkataloge erwiesen hat, durchaus wünschenswert.

- **Hypertexte**
  Hypertexte bieten neben textuellen und audiovisuellen Informationen auch durch die Strukturierung mittels beliebiger Verkettungen ein mächtiges Mittel zur nutzergerechten Informationspräsentation. So dienen sie z.B. als Grundlage von Hilfesystemen zur Benutzerführung oder als HTML-Dateien zur Verbreitung allgemeiner Informationen über das WWW. Ein interessantes Beispiel sind Produktkataloge, die auf Basis des Hilfesystems von MS-Windows die Produktpalette von Firmen des Maschinenbaus präsentieren. Derartige Dateien enthalten nutzbare Informationen im Sinne der oben erwähnten schwach strukturierten Texte. Weiterhin kann es in bestimmten Anwendungsfällen möglich sein, aus der "Verzeigerung" solcher Mengen von Hypertextdateien Metainformationen über Strukturen (z.B. Produktfamilien, Versionen,etc.) oder Zugriffspfade abzuleiten. Die Einbeziehung von HTML-Dateien in Datenbankföderationen ist insbesondere wegen der sehr leistungsfähigen und weitverbreiteten Browser-Applikationen ein wichtiges Thema und Gegenstand vieler aktueller Forschungsaktivitäten.

- **Satzorientierte Dateien**
  Für Dateiformate, die auf Datensätzen gleicher Länge basieren, kann ein effizienter Zugriff über ein

[2] Drawing Exchange Format - zur Abspeicherung geometrischer Informationen im PC Bereich weit verbreitet

FDBS realisiert werden, wenn es gelingt, die in den Applikationsprogrammen verborgene Metainformation zu rekonstruieren. Lediglich die Dateinamen können als Träger von Metainformationen dienen. Der Entwurfsschritt der Datenmodelltransformation kann sich als sehr aufwendig erweisen, da nur Wissen über das Anwendungsszenario zur Verfügung steht und daraus resultierende Vermutungen durch mühsames Probieren überprüft werden müssen. Noch problematischer gestaltet sich dieser Prozeß bei Datensätzen variabler Länge.

- **Binäre Texte**
  Teilweise nutzen Applikationen spezielle Dateiformate um Informationen z.B. als Speicherabzug abzulegen. Solche Dateiformate zu entschlüsseln ist meist nicht oder nur mit sehr großem Aufwand möglich. Auf eine Einbeziehung in die Datenbankföderation muß aus diesem Grunde verzichtet werden.

Die vorgestellten Klassen von Dateiformaten sind weder orthogonal noch vollständig. Weitere Dateiformate, die z.B. Informationen in Form von mehrdimensionalen Tabellen speichern, sind zu berücksichtigen. Somit kann die Zuordnung eines Dateiformates nicht eindeutig erfolgen. Die Klassifizierung kann jedoch als Grundlage für weitere Untersuchungen dienen und wird im Rahmen zukünftiger Arbeiten sukzessive verbessert werden.

# 3 Überlegungen zur Realisierung

Die im Rahmen des Projektes **SIGMA**$_{FDB}$ zu untersuchenden PC-basierten heterogenen Systeme erlauben es im allgemeinen nicht, Einblick in den Programmcode der Spezialapplikationen zu nehmen. Weiterhin stehen zumeist keine allgemeinen wohldefinierten Schnittstellen für den Datenzugriff zur Verfügung. Als gemeinsame Möglichkeit zur Kontrolle von Datenzugriffsoperationen ist die Betriebssystemschnittstelle zum Dateisystem nutzbar.

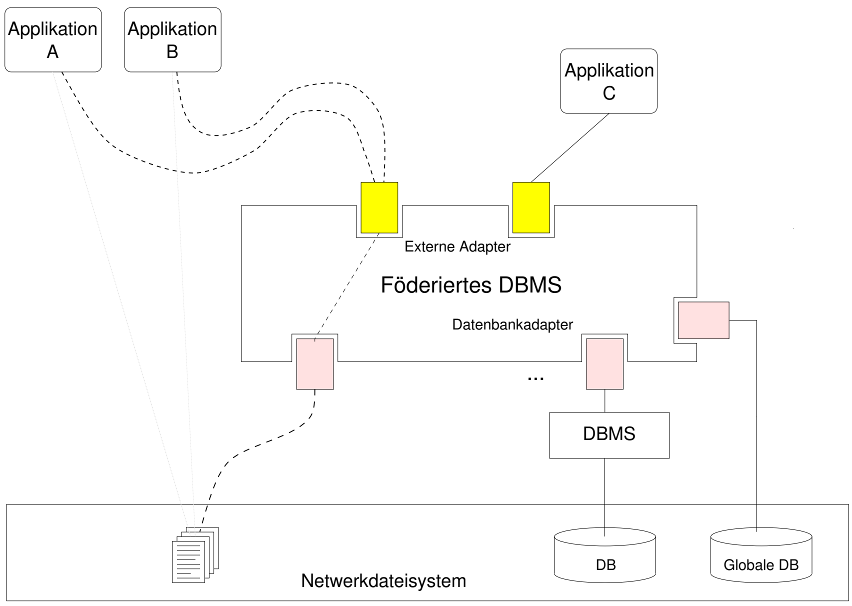


Abbildung 2: Einbindung von dateibasierten Applikationsprogrammen durch Umleitung der Dateizugriffsoperationen auf ein Netzwerkdateisystem

Wird der PC an einem lokalen Netz betrieben und stehen Funktionen eines Netzwerkdateisystems zur Verfügung, so ist die Überwachung der Dateizugriffe durch einen Stellvertreterprozeß möglich. Dazu bietet z.B. das UNIX-File-System die Möglichkeit, low-level Dateioperationen umzuleiten. Realisiert man zum Kern des FDBMS einen externen Adapter (vgl. [Rad94, BK95]), der die gleichen Operationen wie das Netzwerkdateisystem anbietet, so können die Applikationen ohne Änderungen weiterarbeiten. Hierdurch und durch Nutzung eines speziellen Schemakataloges, der die beim föderierten Datenbankentwurfsprozeß gewonnenen bzw. erzeugten Metainformationen enthält[3], kann das FDBMS die Datenzugriffe

[3] z.B. das Mapping zwischen Dateizugriffen und Operationen auf dem äquivalenten Schema oder Zugriffspfade

auf Dateien kontrollieren und gegebenenfalls geeignet reagieren. Über einen lokalen Adapter werden die Dateizugriffsoperationen an das Dateisystem weitergegeben (vgl. Bild 2).

Auf diese Weise können typische, potentiell konsistenzverletzende Zugriffe besonders behandelt werden. Dies sind z.B. schreibende Zugriffe auf Daten einer Datei oder auf Daten, die zwar in verschiedenen Dateien abgespeichert sind, aber dieselbe Information repräsentieren. Solche Daten können durch geeignete Sperrmechanismen und gleichzeitiges Schreiben aller äquivalenten Daten im Rahmen einer Transaktion aktiv konsistent gehalten werden. Eine andere Möglichkeit ist die Konsistenzüberprüfung beim Lesen eines Datums, das gleichzeitig in mehreren Dateien abgelegt ist. Allerdings müssen für dabei offensichtlich werdende Inkonsistenzen geeignete Lösungsstrategien formuliert werden.

Im weiteren wird es möglich, neue Applikationen über weitere externe Adapter auf die föderierten Dateien zugreifen zu lassen bzw. Daten in einem anderen Datenbanksystem zu replizieren. Derartige und andere Fragen sind Gegenstand weiterer Untersuchungen zur Realisierung einer FDBS-Architektur, die auch die Föderierung von speziellen Dateien ermöglicht.

# 4 Zusammenfassung und Ausblick

Die Föderierung von Dateien stellt insbesondere für die im Rahmen des Projektes $\mathbf{SIGMA}_{FDB}$ zu untersuchenden PC-basierten Fabrikplanungstools eine wichtige Aufgabe dar. Um die Frage nach Machbarkeit, Risiko und Kosten einer Föderierung zu beantworten, ist eine Einordnung ähnlich der oben skizzierten Klassifizierung vorzunehmen. Für Dateiformate der beschriebenen Klassen sind geeignete verallgemeinerbare Lösungsstrategien zu entwickeln. Wesentlich ist dabei das Finden bzw. Aufbereiten der Metainformationen, die teilweise in den zu föderierenden Dateien oder teilweise nur in den Applikationsprogrammen enthalten sind. Diese Informationen dienen der Ableitung eines äquivalenten Komponentenschemas und der entsprechenden Mappinginformationen zwischen Dateien und Komponentenschema. Weiterhin kann durch die übernommenen bzw. abgeleiteten Zugriffspfade der effiziente Zugriff unterstützt werden. Die gefundenen und an Beispielapplikationen überprüften Lösungswege sind in geeigneter Weise zu dokumentieren und zu formalisieren. Die mögliche automatische Unterstützung des Entwurfsprozesses, etwa durch Case-Tools, ist zu untersuchen.

Erste Überlegungen zur technischen Realisierung untersuchen die Systemschnittstelle des zu nutzenden Netzwerkdateisystems als gemeinsam genutzte Schnittstelle der verschiedenartigen Datenhaltungssysteme. An dieser Stelle soll das föderierte Datenbankmanagementsystem eingebunden und durch Nutzung der beim Entwurf gefundenen Metainformation dem Gesamtsystem zusätzliche Datenbankfunktionalität hinzugefügt werden, die z.B. den Mehrbenutzerbetrieb unterstützt oder die einfache Realisierung weiterer Applikationen ermöglicht. Dazu sind Fragen über den Aufbau und die effiziente Implementierung von Datenbankadaptern und die geeignete Verwaltung der Metainformationen zu beantworten.

# Literatur

[BK95] R. Böttger und G. Kachel. Implementierungsaspekte der Database Federation Services. In C. Eckert, H.-J. Klein und T. Polle, Herausgeber, *7. GI-Workshop "Grundlagen von Datenbanken"*, S. 25–29. Hildesheimer Informatik Berichte Nr. 12, Juni 1995.

[BLN86] C. Batini, M. Lenzerini und S. B. Navathe. A Comparative Analysis of Methodologies for Database Schema Integration. *ACM Computing Surveys*, 18(4):323–364, Dezember 1986.

[FS94] R. Fietz und C. Schmidt. Weiterentwicklung des Arbeitsplatzes "Logistikgerechte Fabrikplanung". In *Jahresbericht 1994*, S. 42–45. Fraunhoferinstitut für Fabrikbetrieb- und automatisierung Magdeburg, 1994.

[HM85] D. Heimbigner und D. McLeod. A Federated Architecture for Information Management. *ACM Transactions on Office Information Systems*, 3(3):253–278, Juli 1985.

[HTJ+95] M. Höding, C. Türker, S. Janssen, K.-U. Sattler, S. Conrad, G. Saake und I. Schmitt. Föderierte Datenbanksysteme — Grundlagen und Ziele des Projektes $\mathbf{SIGMA}_{FDB}$. Preprint Nr. 12, Fakultät für Informatik, Universität Magdeburg, Dezember 1995.

[LMR90] W. Litwin, L. Mark und N. Roussopoulos. Interoperability of Multiple Autonomous Databases. *ACM Computing Surveys*, 22(3):267–293, September 1990.

[Rad94] E. Radeke. Efendi: Federated Database System of Cadlab. Technical Report 39/94, University of Paderborn & Siemens Nixdorf, Oktober 1994.

[RPRG94] M. P. Reddy, B. E. Prasad, P. G. Reddy und A. Gupta. A Methodology for Integration of Heterogeneous Databases. *IEEE Transactions on Knowledge and Data Engineering*, 6(6):920–933, Dezember 1994.

[Sch95] I. Schmitt. Flexible Integration and Derivation of Heterogeneous Schemata in Federated Database Systems. Preprint Nr. 10, Fakultät für Informatik, Universität Magdeburg, November 1995.

[SL90] A. P. Sheth und J. A. Larson. Federated Database Systems for Managing Distributed, Heterogeneous, and Autonomous Databases. *ACM Computing Surveys*, 22(3):183–236, September 1990.