Теория транзакций

Введение

Базы данных или программы?

- Соотношение данных и программ: время жизни, стоимость, значение.
- Базы данных обеспечивают абстрактные представления данных для всех приложений, использующих эти данные.
- Важнейшее требование к СУБД - поддержка согласованности.
- Данные должны сохраняться в согласованном состоянии независимо от каких-либо ошибок оборудования или программ, отказов, катастроф и т.п.

JET BRAINS RESEARCH

Транзакции

- Транзакция - это совокупность операций, выполняемых прикладной программой, которые переводят согласованное состояние базы данных в согласованное, если:
  - отсутствуют помехи со стороны других приложений;
  - транзакция выполнена полностью.
- Последовательное выполнение транзакций гарантирует сохранение согласованности, но не несовместимо с высокой производительностью системы.
- Система должна обеспечить сохранение согласованности при конкурентном выполнении транзакций, когда операции разных транзакций могут перемежаться.

Приложения, в которых важны транзакционные свойства

- Короткие транзакции в банковских системах, системах резервирования и т.п. (On-line transaction processing - OLTP)
- Распределенные транзакции в системах электронной коммерции
- Координация повторяющихся деловых процедур (потоки работ)

JET BRAINS RESEARCH

Пример некорректного вычисления

t1	t2
read(amt)	
	read(amt)
write (amt+100)	
	write (amt+200)

Согласованность нарушена вследствие недостаточной изоляции транзакций

Пример 2: Перевод между счетами

- Использование высокоуровневых операций:

  Update accounts set bal = bal+amt where id=11

  Update accounts set bal = bal-amt where id=22

- Коммутирующие операции можно выполнять в любом порядке

JET BRAINS RESEARCH

Пример 3: Распределенные системы

1 Покупатель отбирает товар в электронную корзину
2 Перед виртуальной кассой покупатель принимает решение, что в корзине оставить
3 Система запрашивает сведения о покупателе (адрес,

данные о способе оплаты)

4 Система проверяет правильность способа оплаты (авторизация платежа по кредитной карте)
5 Покупатель окончательно подтверждает факт покупки
6 Система инициирует процедуру оплаты
7 Система передает данные на обработку заказа

В транзакции участвует независимая внешняя система (проверка платежного инструмента и оплата покупки

JET BRAINS RESEARCH

Потоки работ

- Обработка заявки на финансирование исследовательского проекта
- Подача заявки
- Регистрация
- Отсылка на экспертизу
- Принятие решения о размере финансирования
- Заключение договора с соискателем

Трехуровневая архитектура распределенных систем

- Клиент: обработка представления для пользователя
- Сервер приложений: обеспечивает хранение и выполнение прикладных программ по запросам клиентов
- Сервер данных: обеспечивает надежное хранение и обработку постоянно хранимых данных по запросам приложений

Функции серверов

- Сервер приложений
  - Прикладные программы могут использовать разделяемые объекты (абстрактные типы данных)
  - Брокер запросов (монитор транзакций)
- Серверы данных
  - Базы данных
  - Серверы документов
  - Почтовые серверы и др.

Более сложные архитектуры

- Специальные случаи: двухуровневые архитектуры
  - Архитектуры клиент-сервер
  - Слияние клиента и сервера приложений: жирные клиенты
  - Слияние сервера приложений с сервером данных: тощие клиенты
- Федерации серверов
  - Клиенты могут посылать запросы к различным серверам приложений.
  - Каждый сервер приложений может посылать запросы нескольким серверам данных.
  - Все серверы автономны и могут быть неоднородными

Реклама: ACID

- Система должна гарантировать маскирование (предотвращение или компенсацию) эффектов, вызванных конкурентным выполнением приложений и любыми отказами
- ACID = Atomicity, Consistency, Isolation, Durability

Атомарность	транзакция либо выполняется полностью, либо не оставляет никаких изменений (фиксация или обрыв)
Согласованность	сохранение согласованного состояния данных
Изолированность	Результаты незавершенной транзакции недоступны для других транзакций
Долговечность	результаты успешно завершенной (зафиксированной) транзакции не могут быть потеряны

JET BRAINS RESEARCH

Вычислительные модели в теории транзакций

Вычислительная модель включает:

- Элементарные операции, определенные над объектами данных
- Транзакции: последовательности или частично упорядоченные множества элементарных операций
- Расписания (или истории), описывающие конкурентное выполнение транзакций
- Критерии корректности расписаний (историй)
- Алгоритмы управления транзакциями, обеспечивающие получение корректных расписаний

JET BRAINS RESEARCH

Страничная модель

- База данных - совокупность независимых и не связанных "страниц": $x, y, \ldots$
- Операции: read, write - пока рассматриваются как атомарные, $r(x), w(x)$
- Полная или частичная упорядоченность операций в транзакции
- Транзакция $t = (\{p\}, <)$ - конечное частично упорядоченное множество шагов $p_i \in \{ r(x_k), w(s_l) \}$, и отношением порядка $<$, такое, что если $p(x)$ и $q(x)$, то $p < q \ \ \mathrm{OR} \ \ q < p$

Семантика страничной модели

Значения, записываемые в операциях записи, зависят от страниц, причитанных на шагах, предшествующих этой операции записи:

$$t = r(x), r(y), w(x), w(y), r(x), r(z), w(x)$$

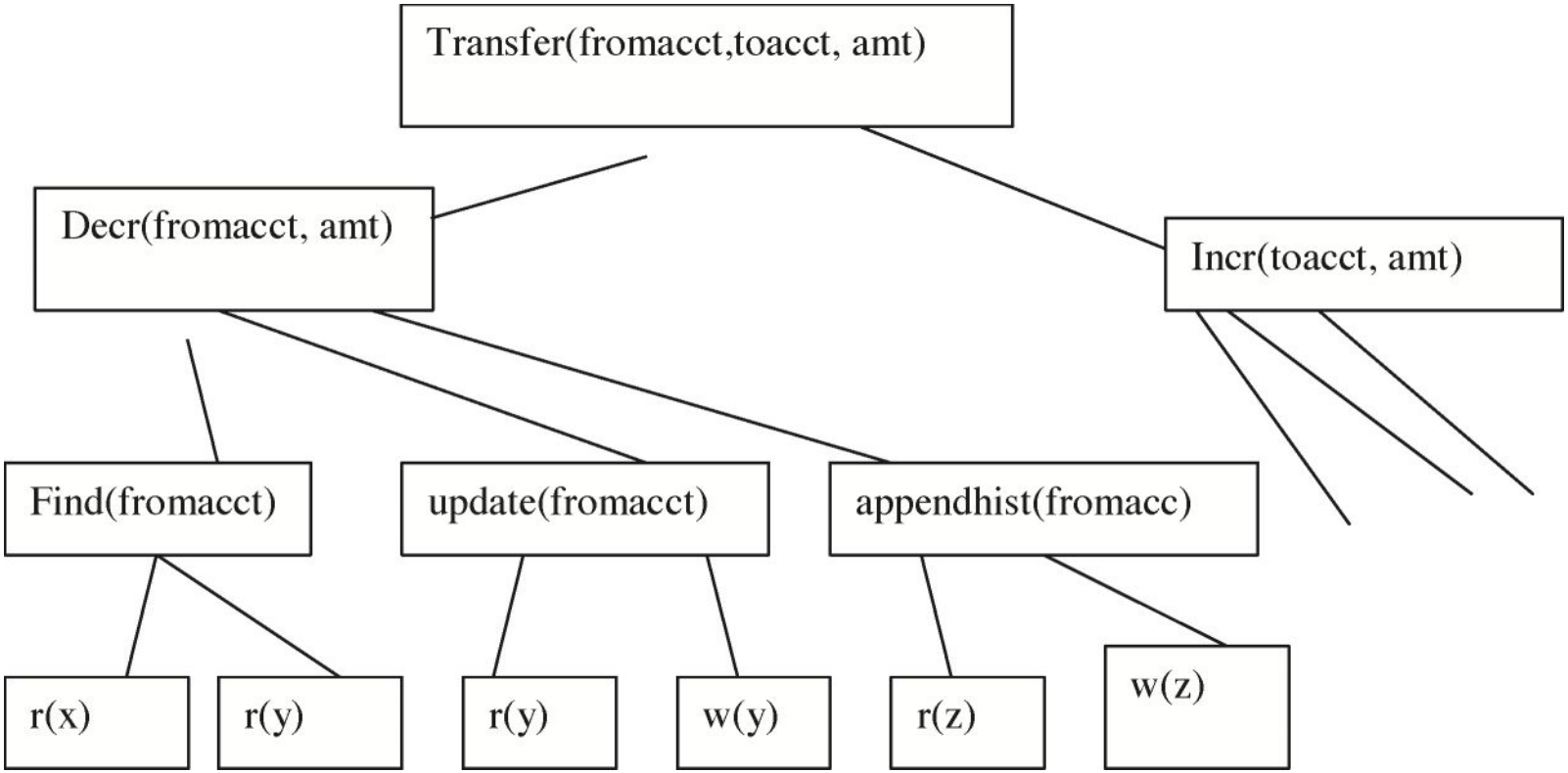
Дополнительные ограничения

- Каждая страница может быть прочитана или записана только один раз
- Никакая страница не читается после того, как была записана

Объектная модель для теории транзакций

- Абстрактные операции над объектами
- Транзакция: конечное дерево с помеченными вершинами
  - Корень помечен идентификатором транзакции
  - Некорневые вершины помечены операциями и параметрами
  - Листья помечены операциями read/write страничной модели.
  - Определен частичный порядок на листьях (как в страничной модели).

Пример транзакции в объектной модели



Теория транзакций

Критерии корректности

Истории и расписания

Операции завершения транзакций: фиксация (commit) c, обрыв (abort) a.
Пусть $T = \{ t_i \}$ - множество транзакций, $t_i = (op_i, <_i)$.
История $s = (op(s), >_s)$, где
$op(s) \subset \cup\, op_i \cup \{ a_i, c_i \}, \quad \cup\, op_i \subset op(s),$
$c_i \in op(s) \Leftrightarrow a_i \notin op(s).$
$\cup <_i \subset <_s \; p_i \in op_i \Rightarrow p_i <_s a_i \vee p_i <_s c_i \,, p_i(x) <_s q_j(x) \vee q_j(x) <_s p_i(x).$
$\cup <_i \subset <_s$
$p_i \in op_i \Rightarrow p_i <_s a_i \vee p_i <_s c_i \,,$
$p_i(x) <_s q_j(x) \vee q_j(x) <_s p_i(x).$

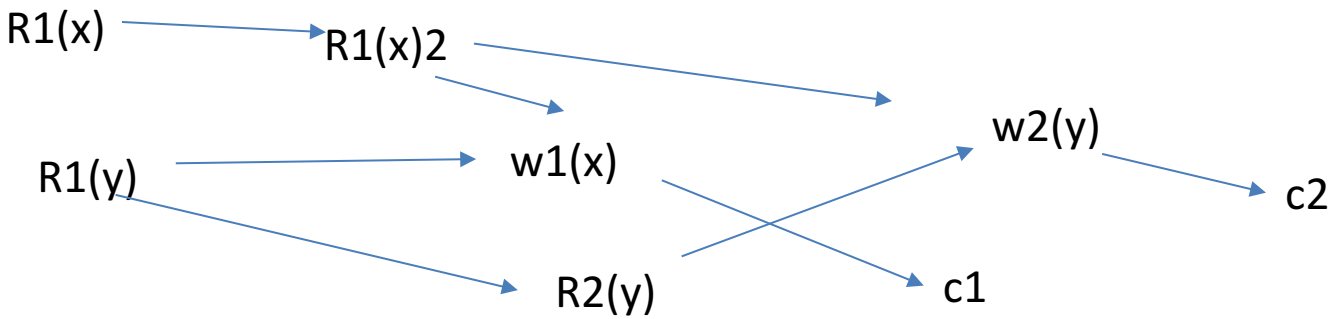
Расписание (Schedule) - префикс истории.

JET BRAINS RESEARCH

Упорядочение операций

- Транзакции

- Расписания



Аномалии конкурентного выполнения

1. Потерянные обновления

   $r_1(x)r_2(x)w_1(x)w_2(x)$

2. Несогласованное чтение

   $r_2(x)w_2(x)r_1(y)r_1(x)r_2(y)w_2(y)$

3. Грязное чтение

   $r_1(x)w_1(x)r_2(x)a_1w_2(x)$

Серийные расписания

- История s называется серийной (serial), если для любых транзакций t_i, $t_j \in trans(s)$ либо все операции t_i предшествуют всем операциям t_j в s, либо все операции t_j предшествуют всем операциям t_i в s,
- Серийное расписание корректно, но неэффективно.

Множества транзакций, связанные с расписанием

- $\mathrm{trans}(s)$ - все транзакции, участвующие в расписании
- $\mathrm{commit}(s)$ - транзакции, успешно завершенные в s.
- $\mathrm{abort}(s)$ - транзакции, оборванные в расписании
- $\mathrm{active}(s)$ - транзакции, незавершенные в расписании

Требования к критериям корректности

- Критерий корректности расписаний должен быть эффективно проверяемым.
- Множество корректных расписаний должно быть достаточно большим.
- Основная идея: определить отношение эквивалентности на множестве расписаний и считать корректными расписания, эквивалентные серийному.
- Оборванные транзакции несущественны для решения вопроса о корректности.

JET BRAINS RESEARCH

Семантика Эрбрана - неформальное описание

- Каждая операция чтения $r_i(x)$ возвращает последнее значение x, записанное предшествующими $w(x)$.
- Операция записи $w_i(x)$ записывает значение, потенциально зависящее от всех значений, прочитанных предшествующими операциями чтения той же транзакции t_i из базы данных или из других транзакций, успешно завершенных или активных в том же расписании.

JET BRAINS RESEARCH

Семантика Эрбрана - операции

Инициализирующая транзакция t_o записывает состояние базы данных и ее завершение предшествует всем операциям других транзакций в расписании.

$$H_s(r_i(x)) = H_s(w_j(x)), \quad i \neq j, w_j(x) < r_i(x), \quad w_k(x) < r_i(x) => w_k(x) < w_j(x)$$

$$H_s(w_i(x)) = f_{ix}(Hs(r_i(y_i)),\ldots\ H_s(r_i(y_m)), r_i(y_k) < w_i(x)$$

Семантики Эрбрана для расписаний

Множество выражений

$$f_{ix}(v_1, \ldots v_m), \qquad i \geq 0, v_k \in HU$$

называется вселенной Эрбрана HU.

Семантикой Эрбрана $\mathrm{H(s)}$ расписания s называется отображение $\mathrm{D} \to \mathrm{HU}$

($\mathrm{D} = \{\mathrm{x}, \mathrm{y}, \ldots.\} -$ база данных)

Сериализуемость по конечному состоянию - FSR

- Определение: Расписания s, s' эквиваленты по конечному состоянию,

$$s \approx_f s',$$
$$\text{если } op(s) = op(s') \wedge H(s) = H(s').$$

- Расписание сериализуемо по конечному состоянию, если оно эквивалентно серийному по конечному состоянию.

Эквивалентность по конечному состоянию - примеры

$r_1(x)r_2(y)w_1(y)r_3(z)w_3(z)r_2(x)w_2(z)w_1(x)$
$r_3(z)w_3(z)r_2(y)r_2(x)w_2(z)r_1(x)w_1(y)w_1(x)$
эквивалентны,

$r_1(x)r_2(y)w_1(y)w_2(y)c_1c_2$	$f_{2y}(f_{0y}())$
$r_1(x)w_1(y)r_2(y)w_2(y)c_1c_2$	$f_{2y}(f_{1y}(f_{0y}()))$

не эквивалентны.

Отношение *читает из*

- Операция p полезна для q, если q транзитивно зависит от p (то есть либо читает из операций, для которых полезна p, либо следует за операциями чтения в той же транзакции, для которых полезна p.

  $RF(s) = \{(t_i, x, t_j)\}$, таких, что $r_j(x)$ читает из $w_i(x)$.

- Операция называется живой, если она полезна для конечного состояния.

$LRF(s) = \{(t_i, x, t_j)\}$, таких, что живая $r_j(x)$ читает из $w_i(x)$.

- Теорема. s, s' - истории, $s \approx_f s'$ если и только если

$$op(s) = op(s') \wedge LRF(s) = LRF(s')$$

Эквивалентность по видимому состоянию

- Желательно, чтобы каждая транзакция в эквивалентных расписаниях читала одно и то же состояние базы данных (view equivalence)
- FSR предотвращает потери обновления, как в

$$r_1(x)r_2(x)w_1(x)w_2(x)c_1c_2$$

но не может предотвратить несогласованное чтение:

$$r_2(x)w_2(x)r_1(x)r_1(y)r_2(y)w_2(y)c_1c_2\,.$$

JET BRAINS RESEARCH

Определение эквивалентности по видимому состоянию

Определение: Расписания s, s' эквивалентны по видимому состоянию $s \approx_v s'$, если

$$op(s) = op(s')$$

$$H(s) = H(s')$$

$$H_s(p) = H_{s'}(p),\ p \in op(s),\ p \text{ — чтение.}$$

Сериализуемость по видимому состоянию - VSR

Теорема (критерий v-эквивалентности):

$$s \approx_v s' \Leftrightarrow D(s) \Leftrightarrow D(s') \Leftrightarrow RF(s) = RF(s')$$

Доказательство: Рассмотрим семантику H и RF для произвольной операции чтения.

Теорема. $\mathrm{VSR} \neq \mathrm{FSR}$.

Доказательство:

$\mathrm{w}_1(\mathrm{x})\mathrm{r}_2(\mathrm{x})\mathrm{r}_2(\mathrm{y})\mathrm{w}_1(\mathrm{y})\mathrm{c}_1\mathrm{c}_2$.

Недостатки VSR

- Никак не помогает устранить аномалии грязного чтения, потому что оборванные транзакции игнорируются.
- Проверка VSR является NP-полной.

Конфликты

Операции различных транзакций находятся в конфликте, если они обрабатывают один и тот же элемент данных и одна из операций является операцией записи.

$\mathrm{con\,f(s)}$ - множество пар конфликтующих операций $\mathrm{p, q} \in \mathrm{s}$, таких, что $\mathrm{p} < \mathrm{q}$.

Расписания s, $\mathrm{s'}$ эквивалентны по конфликтам, $s \approx_c s'$, если

$\mathrm{op(s) = op(s')} \vee \mathrm{con\,f(s) = con\,f(s')}$.

Сериализуемость по конфликтам

- CSR - класс расписаний, эквивалентных по конфликтам серийному.
- Теорема: $\mathrm{CSR} \subset \mathrm{VSR}$.
- Доказательство. Если r_j читает из разных w_i и w_k в s и s', то w_k, w_j находятся в разных конфликтах в этих расписаниях.

$$s = w_1(x)w_2(x)w_2(y)c_2w_1(y)\ c_1w_3(x)w_3(y)c_3$$

Критерий сериализуемости по конфликтам

Граф сериализуемости $\mathrm{G}(\mathrm{s})$:

вершины = $\mathrm{trans}(\mathrm{s})$, направленные дуги связывают транзакции, в которых есть конфликтующие операции.

Теорема. $s \in CSR \Leftrightarrow G(s)$ не имеет контуров.

Коммутативность операций

C1 $r_i(x)r_j(x) \sim r_j(x)r_i(x)$ $\quad i \neq j$

C2 $r_i(x)w_j(y) \sim w_j(y)r_i(x)$ $\quad i \neq j,\ x \neq y$

C3 $w_i(x)w_j(y) \sim w_j(y)w_i(x)$ $\quad i \neq j,\ x \neq y$

C4 $p_i(x), q_j(y)$ неупорядочены $\sim> p_i(x)q_j(y)\ x \neq y \vee (p = r \wedge q = r)$

Расписания эквивалентны по коммутативности, если можно преобразовать одно в другое применением правил C1 — C4.

Эквивалентность по коммутативности совпадает с эквивалентностью по конфликтам.

JET BRAINS RESEARCH

OCSR - сериализуемость с сохранением порядка транзакций

$w_1(x)r_2(x)\ c_2w_3(y)\ c_3w_1(y)c_1 \approx_c t_3t_1t_2$ но $t_2 <_s t_3$

История s называется сериализуемой по конфликтам с сохранением порядка ($OCSR$), если $s \in CSR$ и в эквивалентной серийной истории сохраняется порядок транзакций, строго предшествующих в s.

$OCSR \subset CSR.$

COCSR: Сериализуемость по конфликтам с сохранением порядка фиксации

Расписание $s \in$ COCSR, если для любого конфликта

$$p_i < q_j \Longrightarrow c_i < c_j$$

COCSR $\subset$ CSR.

Доказательство: использовать граф сериализуемости.

$s \in \mathrm{COCSR} \Leftrightarrow s \in \mathrm{CSR}$ и порядок фиксации в эквивалентной серийной истории совпадает с порядком в s.

COCSR $\subset$ OCSR.

Соотношение классов сериализуемости по конфликтам

$$\mathrm{COCSR} \subset \mathrm{OCSR} \subset \mathrm{CSR}$$

$$s_1 = w_1(x)r_2(x)c_2w_3(y)c_3w_1(y)c_1 \in \mathrm{CSR} - \mathrm{OCSR}$$

$$s_2 = w_3(y)c_3w_1(x)r_2(x)c_2w_3(y)c_1 \in \mathrm{OCSR} - \mathrm{COCSR}$$

$$s_3 = w_3(y)c_3w_1(x)r_2(x)w_1(y)c_1c_2 \in \mathrm{COCSR}\text{ - serial}$$

JET BRAINS RESEARCH

Учет обрывов транзакций и отказов системы

Корректность расписаний не должна зависеть от обрывов транзакций и отказов системы.

- $\mathrm{active}(s)$, так как они могут оборваться
- возможен отказ системы, поэтому любой префикс корректного расписания должен быть корректным.

Префиксная замкнутость

$CP(s)$ - проекция расписания s на $commit(s)$.

Класс расписаний называется префиксно-замкнутым, если вместе с каждым расписанием из этого класса любой префикс этого расписания также содержится в этом классе.

Класс расписаний замкнут по фиксациям, если вместе с каждым s он содержит $CP(s)$.

JET BRAINS RESEARCH

FSR - не замкнут, CSR замкнут.

$$s = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1w_3(x)w_3(y)c_3 \approx_v t_1T_2t_3$$

$$s' = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1 \notin \mathrm{FSR}$$

CSR замкнут по фиксациям.

Доказательство: подграф графа (сериализуемости) без контуров не имеет контуров.

Сериализуемость по фиксациям

Определение: Расписание s называется сериализуемым по фиксациям, если $\mathrm{CP}(s')$ для любого префикса s' сериализуемо (в том же смысле).

Теорема:

CMFSR, CMVSR, CMCSR префиксно замкнуты.

$\mathrm{CMCSR} \subset \mathrm{CMVSR} \subset \mathrm{CMFSR}$

$\mathrm{CMFSR} \subset \mathrm{FSR}$

$\mathrm{CMVSR} \subset \mathrm{VSR}$

$\mathrm{CMCSR} = \mathrm{CSR}.$

JET BRAINS RESEARCH

Вложенность классов расписаний

$s_1 = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1$	-FSR
$s_2 = w_1(x)r_2(x)w_2(y)c_2r_1(y)w_1(y)c_1w_3(x)w_3(y)c_3$	FSR-V-CMF
$s_3 = w_1(x)r_2(x)w_2(y)w_1(y)c_1c_2$	CMF+V-CMV
$s_4 = w_1(x)w_2(x)w_2(y)c_2w_1(y)c_1w_3(x)w_3(y)c_3$	VSR-CMF
$s_5 = w_1(x)r_2(x)w_2(y)w_1(y)c_1c_2w_3(x)w_3(y)c_3$	CMFSR-VSR
$s_6 = w_1(x)w_2(x)w_2(y)c_2w_1(y)w_3(x)w_3(y)c_3w_1(z)c_1$	CMVSR-CSR
$s_7 = w_1(x)w_2(x)w_2(y)c_2w_1(z)c_1$	CSR-OCSR
$s_8 = w_3(y)c_3w_1(x)r_2(x)c_2w_1(y)c_1$	OCSR-COCSR
$s_9 = w_3(y)c_3w_1(x)r_2(x)w_1(y)c_1c_2$	COCSR-serial
$s_{10} = w_1(x)w_1(y)c_1w_2(x)w_2(y)c_2$	serial

JET BRAINS RESEARCH